Collected Algorithms from ACM

IN THREE VOLUMES

A collation of all ACM Algorithms, including Certifications, Remarks, and Translations from the Algorithms Department of *Communications of the ACM, 1960–1975*, from *ACM Transactions on Mathematical Software*, 1975 ff, and from *ACM Transactions on Programming Languages and Systems*, 1981 ff.

Introductory Information
Volume III. Algorithms 493ff.

1981

A Service of the Association for Computing Machinery, Inc.
1133 Avenue of the Americas
New York, New York 10036

Quarterly Updating Service Available by Annual Subscription

ACM Algorithms 493ff. are available by purchase in the form of listing, or card deck, or magnetic tape (9 track EBCDIC, 9 track ASCII, 7 track BCD), from ACM Algorithms Distribution Service, c/o International Mathematical & Statistical Libraries, Inc., Sixth Floor, GNB Building, 7500 Bellaire Boulevard, Houston, TX 77036.

Association for Computing Machinery

1133 AVENUE OF THE AMERICAS
NEW YORK, N.Y. 10036
(212) 265-6300

memorandum

acm

December 28, 1981

TO: Subscribers to "Collected Algorithms from ACM"

RE: Supplement No. 80

Attached is the quarterly set of sheets for your loose-leaf volume, "Collected Algorithms from ACM." The sheets in this Supplement provide the material that appeared in the algorithms section of the December 1981 issue of TRANSACTIONS ON MATHEMATICAL SOFTWARE.

The sheets for the new algorithms 578, 579, and 580 should be added to the back of the volume in the following order:

578-P1-0(backed by 578-P2-0)	to	578-P3-0(backed by 578-P4-0)
579-P1-0(backed by 579-P2-0)	to	579-P5-0(backed by blank)
580-P1-0(backed by 580-P2-0)		

The sheets containing the CALGO 20-year Index should be inserted with front matter in the loose-leaf binder immediately following page xiv.

Association for Computing Machinery

11 WEST 42ND STREET
NEW YORK, N.Y. 10036
(212) 869-7440

memorandum

acm

April 26, 1982

TO: Subscribers to "Collected Algorithms from ACM"

RE: Supplement No. 81

Attached is the quarterly set of sheets for your loose-leaf volume, "Collected Algorithms from ACM." The sheets in this Supplement provide the material that appeared in the Algorithms Section of the March 1982 issue of TRANSACTIONS ON MATHEMATICAL SOFTWARE.

The sheets for the new algorithm 581 should be added to the back of the volume in the following order:

581-P1-0(backed by 581-P2-0) to 581-P3-0(backed by 581-P4-0)

The sheets to update algorithms previously provided should be incorporated as follows:

DELETE	INSERT
334-P1-R1(backed by blank)	334-P1-R1(backed by 334-P2-0)

The updated version of the "ACM Algorithms Policy" should replace the one now in the frontmatter, pages ix-xi.

The 1981 Index by Subject to Algorithms should precede the Index by Subject to Algorithms, 1960-1980(pages xv-xxvi).

Association for Computing Machinery

11 WEST 42ND STREET
NEW YORK, N.Y. 10036
(212) 869-7440

memorandum

July 9, 1982

TO: Subscribers to "Collected Algorithms from ACM"

RE: Supplement No. 82

Attached is the quarterly set of sheets for your looseleaf volume, "Collected Algorithms from ACM." The sheets in this Supplement provide the material that appeared in the algorithms section of the June 1982 issue of TRANSACTIONS ON MATHEMATICAL SOFTWARE.

The sheets for the new algorithms 582, 583, and 584 should be added to the back of the volume in the following order:

582-P1-0(backed by 582-P2-0)	to	582-P3-0(backed by 582-P4-0)
583-P1-0(backed by 583-P2-0)	to	583-P11-0(backed by 583-P12-0)
584-P1-0(backed by 584-P2-0)	to	584-P7-0(backed by 584-P8-0)

The sheets to update algorithms previously provided should be incorporated in the looseleaf binders as follows:

DELETE	INSERT
506-P11-0(backed by blank)	506-P11-0(backed by 506-P12-0)
508-P9-0(backed by 508-P10-0)	508-P9-0(backed by 508-P10-R1)
509-P9-0(backed by 509-P10-0)	509-P9-0(backed by 509-P10-R1)

For those with casebound volumes, the editor suggests that you insert the algorithm updates in the volume where the algorithm appears.

Association for Computing Machinery

11 WEST 42ND STREET
NEW YORK, N.Y. 10036
(212) 869-7440

memorandum

acm

October 4, 1982

TO: Subscribers to "Collected Algorithms from ACM"

RE: Supplement No. 83

Attached is the quarterly set of sheets for your looseleaf volume, "Collected Algorithms from ACM." The sheets in this Supplement provide the material that appeared in the algorithms section of the September 1982 issue of TRANSACTIONS ON MATHEMATICAL SOFTWARE.

The sheets for the new algorithms 585, 586, and 587 should be added to the back of the volume in the following order:

585-P1-0(backed by 585-P2-0) to 585-P9-0(backed by 585-P10-0)
586-P1-0(backed by 586-P2-0) to 586-P17-0(backed by blank)
587-P1-0(backed by 587-P2-0) to 587-P9-0(backed by blank)

The sheets to update algorithms previously provided should be incorporated in the looseleaf binders as follows:

DELETE	INSERT
507-P9-0(backed by 507-P10-0)	507-P9-0(backed by 507-P10-R1)

For those with casebound volumes, the editor suggests that you insert the algorithm updates in the volume where the algorithm appears.

Association for Computing Machinery

11 WEST 42ND STREET
NEW YORK, N.Y. 10036
(212) 869-7440

memorandum

acm

January 3, 1983

To: Subscribers to "Collected Algorithms from ACM"

RE: Supplement No. 84

Attached is the quarterly set of sheets for your looseleaf volume, "Collected Algorithms from ACM." The sheets in this Supplement provide the material that appeared in the Algorithms section of the December 1982 issue of TRANSACTIONS ON MATHEMATICAL SOFTWARE.

The sheets for the algorithms 588, 589, 590, and 591 should be added to the back of the volume in the following manner:

588-P1-0(backed by 588-P2-0)		
589-P1-0(backed by 589-P2-0)	to	589-P3-0(backed by 589-P4-0)
590-P1-0(backed by 590-P2-0)	to	590-P5-0(backed by 590-P6-0)
591-P1-0(backed by 591-P2-0)	to	591-P15-0(backed by 591-P16-0)

The sheets to update algorithms previously provided should be incorporated in the looseleaf binders as follows:

DELETE	INSERT
535-P17-0(backed by 535-P18-0)	535-P17-0(backed by 535-P18-R1)
539-31-0(backed by blank)	539-P31-R1(backed by 539-P32-0)
	539-P33-0(backed by blank)
580-P1-0(580-P2-0)	580-P1-0(backed by 580-P2-R1)

The "1982 Index by Subject to Algorithms" should be inserted in front of looseleaf binder with other indexes.

For those with casebound volumes, the editor suggests that you insert the algorithm updates in the volume where the algorithm appears.

Contents

Introductory Information

Introductory Information

Preface

The Algorithms department of *Communications of the ACM* (CACM) was established in February 1960, with J. H. Wegstein as editor, for the purpose of publishing algorithms, consisting of procedures and programs, in the Algol language. In 1975 the publication of ACM algorithms material was transferred to *ACM Transactions on Mathematical Software* (TOMS) and in 1981 *ACM Transactions on Programming Languages and Systems* (TOPLAS) published its first algorithm. A wide variety of algorithms have been published and many of them have been used heavily—either in original form or as translated into other languages. Recognizing the general acceptance of the algorithm material published in CACM, TOMS, and TOPLAS the Association for Computing Machinery (ACM) has collected and reprinted the algorithms to make them more readily accessible and more serviceable to a larger group of users.

The collection contains all algorithms published in the Algorithms departments of CACM, TOMS, and TOPLAS. Covering a great variety of subjects, these algorithms include many for standard computational tasks, such as evaluation of special functions, solution of systems of linear equations, estimation of definite integrals, and sorting of data. Most of them are written in Algol or ANS Fortran.

Both to conserve space and because of changing language usage, reference material on ALGOL and Fortran, [6], [7], [10]–[13], is no longer included as it was in earlier editions. Other expositions of ALGOL are given in [2], [3], [8], and [9]. A new Fortran standard was adopted by ANSI in 1978. The official ANSI document for the new Fortran standard [14] is not yet published, but a summary is given in [4].

A cumulative index to algorithms published since 1960 is provided. In addition to algorithms published in CACM, TOMS, and TOPLAS the index lists many which appeared elsewhere. The classification scheme is a modified form of the SHARE classification. The early indexes were prepared by G. E. Forsythe (Algorithms department editor in 1964) and J. M. Varah, and succeeding editors have annually issued updated versions.

Algorithms 1–220 were originally published as received—without any refereeing whatever. Many of these have since been certified and/or corrected by their authors or by other contributors. Beginning with Algorithm 221, in March 1964, all algorithms have been refereed independently. For many, Certifications have appeared, and modifications to some have been proposed in Remarks. In this volume, Certifications and Remarks for a given algorithm are collected with the algorithm.

Since 1964 an effort has been made to choose for publication those of the refereed algorithms that are interesting and of good quality and, at the same time, likely to be useful to others. Probably few of these algorithms would satisfy all the criteria for excellence proposed by G. E. Forsythe [5]. However, it is hoped that many will be useful and will help to disseminate good methods of solution for many problems. Some of the earlier algorithms which were not refereed are very good too, but those having Certifications and/or corrections (included in Remarks) are more likely to be valuable. It is hoped that users of those without Certifications or Remarks will contribute their experiences to TOMS or TOPLAS so that all may benefit from them. Algorithms 1–50 have been reprinted with revisions and corrections by Ageev, Alik, and Galis in the Soviet Union [1].

For general information, the present Algorithms Policy Statement of TOMS and TOPLAS is provided. The Dissemination Agreement is of particular importance: Submittal of an algorithm for publication in TOMS or TOPLAS implies that unrestricted use of the algorithm within a computer is permissible. General permission to copy the algorithm in fair use, but not for profit, is granted provided ACM's copyright notice is given and reference is made to this publication, its date of issue, and to the fact that copying is by permission of the Association for Computing Machinery.

As new Algorithms, Certifications, Remarks, and Translations appear in TOMS or TOPLAS, they are reissued quarterly to Collected Algorithms subscribers, in loose-leaf form, so that their collections may be kept up to date.

Beginning with Algorithm 493, algorithms are available in machine-readable form—tapes or cards—from ACM Algorithms Distribution Service, c/o IMSL, Sixth Floor, GNB Building, 7500 Bellaire Boulevard, Houston, TX 77063.

To facilitate the updating and to make this volume convenient to use, an understanding of the page numbering scheme for the algorithms is helpful. The page designation is in a three-part format: the left part is the

algorithm number; the middle part is the page number within the algorithm (the first page of each algorithm is P1); and the right part is the number of the revision of that page. All sheets in the original, or first, insertion of an algorithm have "0" for the right part. The first revision of a page will have a page number having the left and middle parts identical with those on the page to be replaced, but the right part will be "R1" instead of "0." The second revision of the same page would read R2, and so on. For example, 123-P2-R1 would mean the first revision of page 2 of Algorithm 123. Revised pages for an algorithm, or additional pages if required, are provided when Certifications or Remarks are added.

The Introduction that appeared in Collected Algorithms at its inception was written by John G. Herriot, department editor at that time. It was then revised by Fred T. Krogh, Algorithms Editor from 1976 to 1978. As present Algorithms Editor, I have prepared this revision to include current information.

Webb Miller
Algorithms Editor
Collected Algorithms from ACM
Department of Mathematics
University of California
Santa Barbara, California 93106

References

1. Ageev, M.A., Alik, V.P., Galis, R.M. Agorithmy 1–50, Akad. Nauk SSSR, Vychislitelnyi Centr, Obschie voprosy programmirovaniya, vypusk 2.
2. Baumann, R., Feliciano, M., Bauer, F.L., and Samelson, K. *Introduction to ALGOL*. Prentice-Hall, Engelwood Cliffs, NJ, 1964.
3. Bottenbruch, H. Structure and Use of ALGOL 60. *J. ACM 9* (Apr. 1962), 161–221.
4. Brainerd, W.S., editor, et al. Fortran 77. *Comm. ACM 10* (Oct. 1978).
5. Forsythe, G.E. Algorithms for Scientific Computation. *Comm. ACM 9* (Apr. 1966), 255–256.
6. Heising, W.P. History and Summary of Fortran Standardization Development for the ASA. *Comm. ACM 7*, (Oct. 1964), 590.
7. Higman, B. What Everybody Should Know About Algol. *Comp. J. 6*, (Feb. 1963), 50–66.
8. McCracken, D.D. *A Guide to Algol Programming*. Wiley, New York, 1962.
9. Schwartz, H.R. An Introduction to Algol. *Comm. ACM 5*, (Feb. 1962), 82–95.
10. Revised Report on the Algorithmic Language Algol 60. *Comm. ACM 6*, (Jan. 1963), 1–17.
11. Fortran vs. Basic Fortran. *Comm. ACM 7* (Oct. 1964), 591–625.
12. Clarification of Fortran Standards—Initial Progress. *Comm. ACM 12* (May 1969), 289–294.
13. Clarification of Fortran Standards—Second Report. *Comm. ACM 14* (Oct. 1971), 628–642.
14. ANSI X3.9-1978 Programming Language FORTRAN. (Revision of X3.9-1960) to appear.

ACM Algorithms Policy

BY FRED T. KROGH[1]

Matters of Content

A contribution should be in the form of an algorithm, comparison, certification, remark, or translation.

Algorithms

ACM Algorithms are published to make the fruits of software research readily available to as wide an audience as possible. An algorithm must either provide a capability not readily available or perform a task better in some way than has been done before. "Better" can mean anything from improved reliability or efficiency to more attractive packaging. In all cases, an algorithm must be of lasting value and must represent a substantial contribution in terms of the amount of work or the originality required for its creation.

In most cases the communication of new algorithmic ideas should be done either in a companion paper published in one of the ACM Transactions or in previously published work. The textual part of an algorithm submission should give a brief description of what the algorithm does and pertinent information on usage and maintenance. It should not duplicate information in another paper or in the algorithm listing. It is recognized that certain algorithms will be useful to readers of one of the ACM Transactions but deal with subjects not in the usual research domain of that journal. When appropriate, the Algorithms Editor will guide authors to other ACM journals for possible publication of such algorithms.

An ACM algorithm must be complete, portable, well documented, and well structured. The meaning of these terms is clarified below.

1. *Completeness:* With the exception of code used from a previously published ACM algorithm, a submission must include all of the code and test data necessary for the effective use and testing of the algorithm by a large segment of its intended audience. To assist those who use the algorithm, a small test driver should be provided that illustrates the use of the algorithm for a simple test problem. For testing purposes, one should provide in a single driver a sufficient variety of test cases to exercise all the main features of the code. All submitted code, including test drivers and preprocessors, is subject to the refereeing process. Code subject to more restricted use than specified in this policy may be used in a supporting role for an algorithm provided it is available from an established source for a nominal fee, and there is no nearly equivalent code available in the public domain.
2. *Portability:* It must be possible to move the code in machine readable form from one machine to another with only minor, well-documented changes. Programs written in a language having several popular dialects should use only language features common to those dialects. As initial evidence of portability, the author can either include evidence of successful execution on three different computers, i.e., computers with different basic instruction sets, or better, when available, can give the result of running the software through a verifier that checks the code for portability or conformance to a standard. All FORTRAN programs must satisfy one of the following criteria:

   (a) Conformance with the ANSI X3.9-1978 FORTRAN 77 [1].

[1] Policy revised March 1979, by Webb Miller; January 1982, by R. J. Hanson.

(b) Pass the set of tests performed by the PFORT verifier [4].[2]

The Algorithms Editor will attempt to help authors who have trouble meeting either of these requirements. The PFORT verifier can be obtained at no cost from the Editor.

Machine-dependent modules, including assembly language, may be used provided:

(a) They are clearly specified, limited in function, small, and either necessary or a substantial improvement over equivalent portable procedures.
(b) Where possible, portable versions are also submitted to facilitate installation and testing, and to confirm the reader's understanding of the specifications. If portable versions are not provided, then tested versions for at least three different computers must be provided.
(c) A portable test program is provided which exhaustively tests each machine-dependent module of the package.

3. *Documentation:* Each module of the code, including test drivers, must be adequately commented. Comments should include the purpose of the module, definitions of all arguments passed through the calling sequence, through global variable declarations, or obtained via input, and comments setting off and explaining major parts of the code. Comments defining machine dependencies should be gathered together and clearly flagged. Standard names and definitions for machine constants are normally to be used [2]. If these constants are used widely in the submitted algorithms, then they may be accessed through a procedure [3]. Other constants clearly identified in comments (e.g., BIG=SQRT (SOVFLO)) and machine dependencies not in the standard [3], may be introduced. A comment can simply point to the place where full comments are given in cases where large comment blocks would otherwise be repeated. An alphabetical list of internal variables together with how they are used is highly recommended. (Identify temporary variables as temporary.) At present, it is not possible to edit comments to provide a reference to the place in the journal where the algorithm appears. Machine-readable documentation giving more detailed information about the package than can be published is encouraged.
4. *Structure:* Code should be organized so that it is easy to use and modify, yet flexible enough to be useful for most problems in the problem area covered. Indentation should be used to identify loops, compound statements and blocks, and continuation of a statement onto another line. If integers are used for statement labels, they must be strictly increasing and should be far enough apart to allow for future modifications. If the space required by an array is highly problem dependent, then that array should have a variable dimension. (NOTE: FORTRAN requires common blocks with the same name to have the same length.)

Algorithmic ideas may be published as integral parts of regular papers without meeting these criteria. This provision may not, however, be used as a subterfuge to circumvent meeting the criteria for ACM algorithms.

Except for extremely long algorithms, all contributions will appear in their entirety in ACM's looseleaf algorithms periodical *Collected Algorithms from ACM* (CALGO). Algorithms are available in machine-readable form through the ACM Algorithms Distribution Service.

---

[2] The PFORT verifier checks a FORTRAN program for conformance with a portable subset of the former ANSI X3.9-1966 FORTRAN standard.

Comparisons

A comparison is a report on the relative merits and features of highly similar software packages for a specific subject area. This study normally includes reporting and interpreting various cogent observed facts about the packages. These facts are typically based on solving a common set of test problems. The drivers, test problems, and data used in the study are to be sent to the Algorithms Editor, at the time of submission, as an algorithm. This body of information will be used in the refereeing process and will be available as an ACM Algorithm.

Certifications

A certification is a report on a previously published algorithm. It can be a careful study of performance characteristics, a verification of correctness, or a report on extensive testing.

Remarks

A remark is a brief report on a previously published algorithm. It is usually concerned with corrections or modifications.

Translations

A translation may either provide machine-dependent modules for a machine or operating system not covered by an ACM algorithm, or a translation of the algorithm into a different high-level language. A translation will only be considered if it is a translation of an algorithm that still represents the current state of the art and it satisfies all of the criteria of a regular algorithm submission.

Submission Formalities

1. Five copies of all textual material should be sent to the Algorithms Editor. This material must be typewritten and double spaced. Companion papers to algorithms submitted to one of the ACM Transactions should be sent with the algorithm to the Algorithms Editor, as they will be refereed along with the algorithm. The purpose of those parts of the code submission that are included for testing purposes or serve only a supporting role should be clearly identified.
2. Processing of an algorithm will not proceed until it is verified that machine-readable copy of the algorithm has arrived in good form. Before sending machine-readable copy, contact the Algorithms Editor and explain the computer(s) available to you, the size of your submission, and in what form you would like to send the material. If an algorithm requires the use of code subject to restricted use, the use of this code should be cleared first with the Algorithms Editor.
3. Evidence of portability must be included with an algorithm submission.

Dissemination Agreement

Submittal of an algorithm for publication in one of the *ACM Transactions* implies that unrestricted use of the algorithm within a computer is permissible. General permission to copy and distribute the algorithm without fee is granted provided that the copies are not made or distributed for direct commercial advantage. The ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

Copyright Agreement

Authors of copyrightable algorithms (or their employers) are required to transfer the copyright to ACM upon acceptance of the algorithm for publication, in accordance with ACM policy to own copyright on ACM published material. The Association grants to authors the right to reuse their material and seeks the permission of authors before approving republication.

REFERENCES

1. AMERICAN NATIONAL STANDARDS INSTITUTE. American National Standard programming language FORTRAN, ANSI X3.9-1978, American National Standards Institute, New York, 1978.
2. FORD, B. Parameterization of the environment for transportable numerical software. *ACM Trans. Math. Softw. 4*, 2 (June 1978), 100-103.
3. FOX, P. A., HALL, A. D., AND SCHRYER, N. L. Algorithm 528: Framework for a portable library. *ACM Trans. Math. Softw. 4*, 2 (June 1978), 177-188.
4. RYDER, B. F., AND HALL, A. D. The PFORT verifier. Bell Laboratories Computer Science Rep. 2 (May 1973-January 1981).

Software Package Policy

The ACM Algorithm Distribution Service makes it possible to remove the length limitation on algorithms published in TOMS and TOPLAS and allows them to publish and disseminate large programs.

Technically, a collection of algorithms is just another algorithm, but practically ACM editors will distinguish between "ordinary" algorithms and larger entities called polyalgorithms, software packages, systemized collections, etc. This policy statement is an extension of the algorithms policy of TOMS and TOPLAS which describes those software packages that these Transactions are interested in publishing and gives the specific requirements expected for such packages.

General Criteria and Procedures

Usefulness is an important criterion for the acceptance of any ACM algorithm; it is an essential criterion for ACM software packages. An acceptable ACM package must address itself to a standard and widely occurring problem set. Furthermore, it must be designed to ease the user's task as much as possible consistent with the nature of the problem set. The human engineering aspects of the package will be weighted heavily in its evaluation by the editors.

All ACM algorithms, but especially packages, are expected to be of high quality in all respects. ACM packages must use good algorithms in their components, they must use good programming (including style, structure, and adherence to standards), they must be well documented (both for use and modification), and they must be thoroughly tested and evaluated.

Transportability will also be weighted heavily in that it directly affects usefulness. A wonderful package will not be accepted if it requires a specific environment of machine, operating system and I/O devices that is available only at a very few locations. However, complete transportability will not be insisted upon, and in each case a judgment will be made by the editors on whether a reasonable and useful level of transportability has been achieved. It is widely realized that, in general, high level language algorithms cannot yet be constructed to give top performance in a variety of environments.

ACM publication normally consists of two parts: a description of the package and a critical review of it. Authors of a package will submit material for the first part and the editors will seek referee/reviewers for the second part. We note that a multitude of widely distributed software packages already exist, some of excellent quality. We consider such packages to have already been published. Other packages may become widely distributed in the future. Authors independent of the developers of these packages are invited to submit critical reviews of them. An author of a new package is expected to demonstrate the superiority of the new package over any existing packages.

Specific Requirements

The following guidelines for authors give information about the expected contents of papers to be printed in TOMS or TOPLAS, the nature of the packages to be distributed, and the material to be submitted for consideration by the editors.

(1) *The Descriptive Paper.* The paper should contain a clear statement of goals, intended users, expected environment, and the problem domain of the package. Then there should be a discussion of the main components and steps in the development of the package. This includes mathematical descriptions, general algorithm structure, testing and evaluation, documentation, programming, and portability. Interesting design questions should be discussed here. It is not expected that a complete mathematical derivation be given; rather one should give pertinent references. Descriptions of new techniques belong in an independent paper. The aim here is to give the reader a general and reasonably complete description without burdening him with the full details of documentation and techniques. Full details are available from the references and the ACM Algorithm Distribution Service. The paper should give some basic statistics about the package, e.g. length, examples of performance (memory usage, execution times), I/O requirements.

This policy was originally written for TOMS in 1975 by Editor-in-Chief John R. Rice, with Richard M. Heiberger and Charles L. Lawson, who made substantial contributions to the formulation of the policy, and with Edward Battiste, David C. Hoaglin, and Paul Velleman, who made valuable suggestions. This policy now applies also to TOPLAS, in which the first algorithm was published in 1981.

(2) *The Package.* The ACM Algorithm Distribution Service makes three items available: the programs, user documentation, and internal documentation. All documentation is incorporated into the program as machine-readable comments. A package must be of acceptable quality in all of the following aspects: user documentation; internal documentation; ease of use; programming style; consistency of style, structure, and notation; basic algorithmic and mathematical techniques; adherence to language standards; ease of modification and transportation; performance characteristics; overall structure and organization.

(3) *Material To Be Submitted.* The material for an ACM software package should be submitted to the Algorithms Editor or to the Editor-in-Chief of TOMS or of TOPLAS and should consist of the following items:

—machine readable form of package
—descriptive paper (three copies)
—documentation (user and internal—probably part of first item)
—drivers for examples and testing
—test and evaluation results (listings from actual runs, summaries, etc.).

The drivers submitted should exercise all the modules in the package. The test and evaluation results must support the claim of superiority over existing packages; a separate discussion of this aspect will probably be required.

It must be emphasized that finding referees for software packages is likely to be difficult, and that if competent referees cannot be found it will be necessary to reject the manuscript. Authors will improve their chances of finding willing referees if they submit their material in a neat and well organized form. All computer tests submitted should be carefully annotated so that the referee can easily determine the nature of the test, the test data used, the expected results, and the results produced by the package.

(4) *The Critical Review.* This paper will analyze the strengths and weaknesses of the package in an objective manner. It may include extensive discussions of only a few aspects of the package if they are of sufficient interest (particularly for already existing packages). Critical reviews of widely disseminated packages not published in either TOMS or TOPLAS must contain basic information about their distribution, listings, and maintenance support.

Editorial Comment

It is clear that the preparation of a publishable software package is a task well beyond the usual technical publication. The critical review part of publication is a way in which referees can obtain suitable recognition for the substantial effort involved in a thorough evaluation of a package.

1982 Index by Subject to Algorithms

Algorithms published elsewhere are included here. A cumulative index to algorithms published in *Communications*, the *ACM Transactions on Mathematical Software*, and the *ACM Transactions on Programming Languages and Systems* from 1960 through 1980 may be consulted in "Collected Algorithms from ACM," which is a looseleaf collection of all algorithms, each with certifications and remarks, if any, appended. The 1981 index appears in the December 1981 issue of *Communications*.

QUADRATURE [D1].

Laurie, D.P. Algorithm 584: CUBTRI—Automatic cubature over a triangle. (Fortran) *ACM Trans. Math. Softw. 8*, 2 (June 1982), 210–218.

Robinson, I., and deDoncker, E. Algorithm 45: Automatic computation of improper integrals over a bounded or unbounded planar region. (Fortran) *Computing 27*, 3 (1981), 253–284.

INTEGRAL EQUATIONS [D5]

Piessens, R. Algorithm 113: Inversion of the Laplace transform (ALGOL) *Computer Journal 25*, 2 (May 1982), 278–282.

INTERPOLATION [E1]

Brezinski, C. Algorithm 585: A subroutine for the general interpolation and extrapolation problems. (Fortran) *ACM Trans. Math. Softw. 8*, 3 (Sept. 1982), 290–301.

Hanson, R.J. Remark on Algorithm 507: Procedures for quintic natural spline interpolation. *ACM Trans. Math. Softw. 8*, 3 (Sept. 1982), 334.

MATRIX OPERATIONS [F1]

Chan, T.F. Algorithm 581: An improved algorithm for computing the singular value decomposition. (Fortran) *ACM Trans. Math. Softw. 8*, 1 (March 1982), 84–88.

Lewis, J.G. Algorithm 582: The Gibbs-Poole-Stockmeyer and Gibbs-King algorithms for reordering sparse matrices. (Fortran) *ACM Trans. Math. Softw. 8*, 2 (June 1982), 190–194.

Mönch, W. Algorithm 46: Iterative refinement of approximations to a generalized inverse of a matrix. (ALGOL) *Computer 28*, 1 (1982), 79–87.

Dodson, D.S., and Grimes, R.G. Remark on Algorithm 539: Basic linear algebra subprograms for Fortran Usage. *ACM Trans. Math. Softw. 8*, 4 (Dec. 1982), 403–404.

Lewis, J. G. Remark on Algorithms 508 and 509: Matrix bandwidth and profile reduction; A hybrid profile reduction algorithm. *ACM Trans. Math Softw. 8*, 2 (June 1982), 221.

EIGENVALUES AND EIGENVECTORS OF MATRICES [F2]

Dongarra, J.J. Algorithm 589: SICEDR: A Fortran subroutine for improving the accuracy of computed matrix eigenvalues. *ACM Trans. Math. Softw. 8*, 4 (Dec. 1982), 371–375.

van Dooren, P. Algorithm 590: DSUBSP and EXCHQZ: Fortran subroutines for computing deflating subspaces with specified spectrum. *ACM Trans. Math. Softw. 8*, 4 (Dec. 1982), 376–382.

Garbow, B.S. Remark on Algorithm 535: The QZ algorithm to solve the generalized eigenvalue problem. *ACM Trans. Math. Softw. 8*, 4 (Dec. 1982), 402.

Flamm, D.S., and Walker, R.A. Remark on Algorithm 506. HQR3 and EXCHNG: Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix. *ACM Trans. Math. Softw. 8*, 2 (June 1982), 219–220.

SIMULTANEOUS LINEAR EQUATIONS [F4]

Paige, C.C., and Saunders, M.A. Algorithm 583: LSQR—Sparse linear equations and least squares problems. (Fortran) *ACM Trans. Math. Softw. 8*, 2 (June 1982), 195–209.

Kincaid, D.R., Respess, J.R., and Young, D.M. Algorithm 586: ITPACK 2C—A Fortran package for solving large sparse linear systems by adaptive accelerated iterative methods. (Fortran) *ACM Trans. Math. Softw. 8*, 3 (Sept. 1982), 302–322.

Hanson, R.J., and Haskell, K.H. Algorithm 587: Two algorithms for the linearly constrained least squares problem. (Fortran) *ACM Trans. Math. Softw. 8*, 3 (Sept. 1982), 323–333.

ORTHOGONALIZATION [F5]

Buckley, A. Remark on Algorithm 580: GRUP: A set of Fortran routines for updating QR factorizations. *ACM Trans. Math. Softw. 8*, 4 (Dec. 1982), 405.

SIMPLE CALCULATIONS ON STATISTICAL DATA [G1]

Hemmerle, W.J. Algorithm 591: A comprehensive, matrix-free algorithm for analysis of variance. *ACM Trans. Math. Softw. 8*, 4 (Dec. 1982), 403–404.

Frome, E.L., Algorithm 171: Fischer's exact variance test for the Poisson distributions. (Fortran) *Applied Statistics 31*, 1 (1982), 67–71.

MacKenzie, G., and O'Flaherty, M. Algorithm 173: Direct design matrix generation for balanced factorial experiments. (Fortran) *Applied Statistics 31*, 1 (1982), 74–80.

Schwertman, N.C. Algorithm 174: Multivariate multisample non-parametric tests. (Fortran) *Applied Statistics 31*, 1 (1982), 80–85.

Laurie, D.P. Algorithm 175: Cramer-Wold factorization. (Fortran) *Applied Statistics 31*, 1 (1982), 86–93.

Silverman, B.W. Algorithm 176: Kernel density estimation using the Fast Fourier Transform. (Fortran) *Applied Statistics 31*, 1 (1982), 93–99.

RANDON NUMBER GENERATORS [G5]

Tracht, A.E. Remark on Algorithm 334: Normal random deviates. *ACM Trans. Math. Softw. 8*, 1 (March 1982), 89.

OPERATIONS RESEARCH, GRAPH STRUCTURES [H]

Carpanto, G., and Toth, P. Algorithm 44: Algorithm for the solution of the bottleneck assignment problem. (Fortran) *Computing 27*, 2 (1981), 179–187.

Fayard, D., and Plateau, G. Algorithm 47: An algorithm for the solution of 0-1 knapsack problem. (Fortran) *Computing 28*, 3 (1982), 269–287.

APPROXIMATION OF SPECIAL FUNCTIONS [S17]

Anderson, W.L. Algorithm 588: Fast Hankel transforms using related and lagged convolutions. *ACM Trans. Math. Softw. 8*, 4 (Dec. 1982), 369–370.

ALL OTHERS [Z]

O'Flaherty, M., and MacKenzie, G. Algorithm 172: Direct simulation of nested Fortran DO-loops. (Fortran) *Applied Statistics 31*, 1 (1982), 71–74.

Frost, R.A. Algorithm 112: Dumping the index of a dynamic hash table. (ALGOL 68-R) *Computer Journal 24*, 4 (Nov. 1981), 383–384.

1981
Index by Subject to Algorithms

Algorithms published elsewhere are included. A cumulative index to algorithms published in *Communications*, the ACM *Transactions on Mathematical Software*, and the *ACM Transactions on Programming Languages and Systems* from 1960 through 1980 may be consulted in "Collected Algorithms from ACM," which is a looseleaf collection of all algorithms, each with certifications and remarks, if any, appended.

REAL ARITHMETIC; NUMBER THEORY [A1].
OPERATIONS ON POLYNOMIALS AND POWER SERIES [C1]

Lozier, D.W., and Smith, J.M. Algorithm 567: Extended-range arithmetic and normalized Legendre polynomials. (Fortran) *ACM Trans. Math. Softw. 7*, 1 (March 1981), 141–146.

ZEROS OF ONE OR MORE NONLINEAR EQUATIONS [C5]

More, J.J., Garbow, B.S., and Hillstrom, K.E. Algorithm 566: Fortran subroutines for testing unconstrained optimization software. (Fortran) *ACM Trans. Math. Softw. 7*, 1 (March 1981), 136–140.

Incerti, S., Zirilli, F., and Parisi, V. Algorithm 111: A Fortran subroutine for solving systems of nonlinear, simultaneous equations. (Fortran) *Computer Journal 24*, (1981), 87–91.

ORDINARY DIFFERENTIAL EQUATIONS [D2]

Ascher, U., Christiansen, J., and Russell, R.D. Algorithm 569: COLSYS—Collocation software for boundary-value ODE's. (Fortran) *ACM Trans. Math. Softw. 7*, 2 (June 1981), 223–229.

PARTIAL DIFFERENTIAL EQUATIONS [D3]

Melgaard, D.K., and Sincovec, R.F. Algorithm 565: PDETWO/PSETM/GEARB—Solution of systems of two dimensional nonlinear partial differential equations. (Fortran) *ACM Trans. Math. Softw. 7*, 1 (March 1981), 126–135.

O'Leary, D.P., and Widlund, O. Algorithm 572: Solution of the Helmholtz equation for the Dirichlet problem on general bounded three-dimensional regions. (Fortran) *ACM Trans. Math. Softw. 7*, 2 (June 1981), 239–246.

DIFFERENTIATION [D4]

Fornberg, B. Algorithm 579: CPSC: complex power series coefficients. (Fortran) *ACM Trans. Math. Softw. 7*, 4 (Dec. 1981), 542–547.

INTERPOLATION [E1]
CURVE AND SURFACE FITTING [E2]

McAllister, D.F., and Roulier, J.A. Algorithm 574: Shape-preserving osculatory quadratic splines. (Fortran) *ACM Trans. Math. Softw. 7*, 3 (Sept. 1981), 384–386.

MINIMIZING OR MAXIMIZING A FUNCTION [E4]

More, J.J., Garbow, B.S., and Hillstrom, K.E. Algorithm 566: Fortran subroutines for testing unconstrained optimization software. (Fortran) *ACM Trans. Math. Softw. 7*, 1 (March 1981), 136–140.

Dennis, J.E., Jr., Gay, D.M., and Welsch, R.E. Algorithm 573: NL2SOL—An adaptive nonlinear least-squares algorithm. (Fortran) *ACM Trans. Math. Softw. 7*, 3 (Sept. 1981), 369–383.

MATRIX OPERATIONS, INCLUDING INVERSION [F1]

Duff, I.S. Algorithm 575: Permutations for a zero-free diagonal. (Fortran) *ACM Trans. Math. Softw. 7*, 3 (Sept. 1981), 387–390.

EIGENVALUES AND EIGENVECTORS OF MATRICES [F2]

Stewart, W.J., and Jennings, A. Algorithm 570: LOPSI—A simultaneous iteration algorithm for real matrices. (Fortran) *ACM Trans. Math. Softw. 7*, 2 (June 1981), 230–232.

SIMULTANEOUS LINEAR EQUATIONS [F4]

Barrodale, I., and Stuart, G.F. Algorithm 576: A Fortran program for solving Ax = b. (Fortran) *ACM Trans. Math. Softw. 7*, 3 (Sept. 1981), 391–397.

DuCroz, J.J., Nugent, S.M., Reid, J.K., and Taylor, D.B. Algorithm 578: Solution of real linear equations in a paged virtual store. (Fortran) *ACM Trans. Math. Softw. 7*, 4 (Dec. 1981), 537–541.

ORTHOGONALIZATION [F5]

Buckley, A. Algorithm 580: QRUP; a set of Fortran routines for updating QR factorizations. (Fortran) *ACM Trans. Math. Softw. 7*, 4 (Dec. 1981), 548–549.

SIMPLE CALCULATIONS ON STATISTICAL DATA [G1]

Gardner, G., Harvey, A.C., and Phillips, G.D.A. Algorithm AS 154: An algorithm for exact maximum likelihood estimation of autoregressive-moving average models by means of Kalman filtering. (Fortran) *Applied Statistics 29* (1980), 311–322.

Davis, R. Algorithm AS 155: The distribution of a linear combination of x^2 random variables. (Algol) *Applied Statistics 29* (1980), 323–333.

Jones, B. Algorithm AS 156: Combining two component designs to form a row-and-column design. (Fortran) *Applied Statistics 29* (1980), 334–337.

Grafton, R. Algorithm AS 157: The runs-up and runs-down tests. (Fortran) *Applied Statistics 30* (1981), 81–85.

Cran, G. Algorithm AS 158: Calculation of the probabilities {P(I,k)} for the simply ordered alternative. (Fortran) *Applied Statistics 30* (1981), 85–91.

Patefield, W. Algorithm AS 159. An efficient method of generating random $R \times C$ tables with given row and column totals. (Fortran) *Applied Statistics 30* (1981), 91–97.

Lusbader, E., and Stodola, R. Algorithm AS 160: Partial and marginal association in contingency tables. (Fortran) *Applied Statistics 30* (1981), 97–105.

APPROXIMATION OF SPECIAL FUNCTIONS [S]

Hill, G.W. Algorithm 571: Statistics for von Mises' and Fischer's distributions of directions—$I_1(x)/I_0(x)$, $I_{1.5}(x)/I_{0.5}(x)$, and their inverses. (Fortran) *ACM Trans. Math. Softw. 7*, 2 (June 1981), 233–238.

Carlson, B.C., and Notis, E.M. Algorithm 577: Algorithms for incomplete elliptic integrals. (Fortran) *ACM Trans. Math. Softw. 7*, 3 (Sept. 1981), 398–403.

Hill, G.W. Remark on Algorithm 395: Student's *t*-distribution. *ACM Trans. Math. Softw. 7*, 2 (June 1981), 247–249.

Hill, G.W. Remark on Algorithm 396: Student's *t*-quantiles. *ACM Trans. Math. Softw. 7*, 2 (June 1981), 250–251.

Razaz, M., and Schonfelder, J.L. Remark on Algorithm 498: Airy functions using Chebyshev series approximations. *ACM Trans. Math. Softw. 7*, 3 (Sept. 1981), 404–405.

ALL OTHERS [Z]

Hanson, D.R. Algorithm 568: PDS—A portable directory system. (Ratfor and Fortran) *ACM Trans. Program. Lang. Syst. 3*, 2 (April 1981), 162–167.

Index By Subject To Algorithms, 1960–1980

Algorithms are grouped according to the classification system below, which is a modification of the SHARE classification system. Algorithms published elsewhere are included; see list of abbreviations for other sources.

Each entry gives the key to the classification system categories (A1, B1, . . .); the number of the algorithm if it exists; abbreviated title with language designator (A for Algol, F for Fortran, etc.—these appear on the more recent algorithms); source designator, volume, year, page (in parenthesis).

Classification System

Z All Others
S Approximation of Special Functions
G6 Combinations and Permutations
L2 Compiling
A2 Complex Arithmetic
I5 Composite Input
O2 Computing Structure Simulation
C6 Convergence Acceleration
M2 Conversion and Scaling of Data
G2 Correlation and Regression Analysis
E2 Curve and Surface Fitting
M2 Data Conversion and Scaling
F3 Determinants
D2 Differential Equations, Ordinary
D3 Differential Equations, Partial
D4 Differentiation
F2 Eigenvalues and Eigenvectors of a Matrix
B3 Exponential and Logarithmic Functions
E4 Function Minimizing and Maximizing
H Graph Structures, Operations Research
B2 Hyperbolic Functions
I5 Input—Composite
D5 Integral Equations
E1 Interpolation
F1 Inversion of a Matrix
F4 Linear Equations, Simultaneous
B3 Logarithmic and Exponential Functions
F2 Matrix Eigenvalues and Eigenvectors
F1 Matrix Operations, Including Inversion
F3 Matrix, Determinant of
E4 Maximizing a Function
E4 Minimizing a Function
C5 Nonlinear Equations, Zeroes of
A1 Number Theory
C1 Operations on Polynomials and Power Series
H Operations Research, Graph Structures
D2 Ordinary Differential Equations
F5 Orthogonalization
D3 Partial Differential Equations
G6 Permutations and Combinations
Y1 Physics Applications
J6 Plotting
C1 Polynomials, Operations on
C2 Polynomials, Zeroes of
C1 Power Series, Operations on
B4 Powers and Roots
D1 Quadrature
G5 Random Number Generators
A1 Real Arithmetic
G2 Regression and Correlation
K2 Relocation
B4 Roots and Powers
MS Scaling and Conversion of Data
C6 Series, Summation and Convergence Acceleration of
G1 Simple Calculations on Statistical Data
O2 Simulation of Computing Structure
F4 Simultaneous Linear Equations
E3 Smoothing
M1 Sorting
S Special Functions, Approximation of
G2 Statistical Data, Correlation and Regression Analysis of
G1 Statistical Data, Simple Calculations on
G7 Subset Generators
C6 Summation of Series
E2 Surface and Curve Fitting
R2 Symbol Manipulation
B1 Trig and Inverse Trig Functions
F5 Vectors, Orthogonalization of
C5 Zeroes of one or more Nonlinear Equations
C2 Zeroes of Polynomials

Classification of Special Functions

S04 Bernoulli and Euler Numbers and Polynomials
S18 Bessel Function, Modified
S19 Bessel Functions of Complex Argument
S18 Bessel Functions of Pure Imaginary Argument
S17 Bessel Functions of Real Argument
S20 Bessel and Related Functions, Miscellaneous
S14 Beta Function and Incomplete Beta Function
S03 Binomial Coefficients
S07 Circular Functions, Miscellaneous
S19 Complex Argument, Bessel Functions of
S13 Cosine Integrals
S23 Curve-Fitting
S04 Derivatives and Differences of Zero
S15 Derivatives
S04 Differences and Derivatives of Zero
S23 Differentiation, Numerical
S21 Elliptic Integrals and Functions
S15 Error Integral
S04 Euler and Bernoulli Numbers and Polynomials
S13 Exponential Integrals
S14 Factorial Function
S03 Factorials
S22 Functions: Miscellaneous Higher Mathematical Functions
S14 Gamma Function and Incomplete Gamma Function
S15 Hermite Polynomials and Functions
S15 Higher Integrals
S22 Higher Mathematical Functions, Miscellaneous
S18 Imaginary Argument, Bessel Functions of
S14 Incomplete Beta and Gamma Functions
S13 Integrals of Exponentials, Logarithms, Sines, Cosines, etc.
S21 Integrals, Elliptic

CLASSIFICATION SYSTEM (MODIFIED SHARE)

A1 REAL ARITHMETIC, NUMBER THEORY
A2 COMPLEX ARITHMETIC
B1 TRIG AND INVERSE TRIG FUNCTIONS
B2 HYPERBOLIC FUNCTIONS
B3 EXPONENTIAL AND LOGARITHMIC FUNCTIONS
B4 ROOTS AND POWERS
C1 OPERATIONS ON POLYNOMIALS AND POWER SERIES
C2 ZEROS OF POLYNOMIALS
C5 ZEROS OF ONE OR MORE NON-LINEAR EQUATIONS
C6 SUMMATION OF SERIES, CONVERGENCE ACCELERATION
D1 QUADRATURE
D2 ORDINARY DIFFERENTIAL EQUATIONS
D3 PARTIAL DIFFERENTIAL EQUATIONS
D4 DIFFERENTIATION
D5 INTEGRAL EQUATIONS
E1 INTERPOLATION
E2 CURVE AND SURFACE FITTING
E3 SMOOTHING
E4 MINIMIZING OR MAXIMIZING A FUNCTION
F1 MATRIX OPERATIONS, INCLUDING INVERSION
F2 EIGENVALUES AND EIGENVECTORS OF MATRICES
F3 DETERMINANTS
F4 SIMULTANEOUS LINEAR EQUATIONS
F5 ORTHOGONALIZATION
G1 SIMPLE CALCULATIONS ON STATISTICAL DATA
G2 CORRELATION AND REGRESSION ANALYSIS
G5 RANDOM NUMBER GENERATORS
G6 PERMUTATIONS AND COMBINATIONS
G7 SUBSET GENERATORS
H OPERATIONS RESEARCH, GRAPH STRUCTURES
I5 INPUT - COMPOSITE
J6 PLOTTING
K2 RELOCATION
L2 COMPILING
M1 SORTING
M2 DATA CONVERSION AND SCALING
O2 SIMULATION OF COMPUTING STRUCTURE
R2 SYMBOL MANIPULATION
S APPROXIMATION OF SPECIAL FUNCTIONS...
S FUNCTIONS ARE CLASSIFIED SØ1 TO S22, FOLLOWING
S FLETCHER-MILLER-ROSENHEAD, INDEX OF MATH. TABLES
Y1 PHYSICS APPLICATIONS
Z ALL OTHERS

ABBREVIATIONS FOR SOURCES

AL ARGONNE NATIONAL LABORATORY, NATS PROJECT
AS APPLIED STATISTICS
BT B.I.T.
CA COMMUNICATIONS OF THE ASSOCIATION FOR COMPUTING MACHINERY
CB COMPUTER BULLETIN
CC COMPUTER PHYSICS COMMUNICATIONS
CH CHIFFRES
CJ COMPUTER JOURNAL
CP COMPUTING
HA U.K.A.E.A., HARWELL, REPORTS
IC I.C.C. BULLETIN (INTERNATIONAL COMPUTATION CENTRE)
JC JOURNAL OF THE ASSOCIATION FOR COMPUTING MACHINERY
MC MATHEMATICS OF COMPUTATION
NM NUMERISCHE MATHEMATIK
SN SIAM JOURNAL ON NUMERICAL ANALYSIS
TO TRANSACTIONS ON MATHEMATICAL SOFTWARE
ZM ZASTOSOWANIA MATEMATYKI

LANGUAGE CODE

A ALGOL
B BASIC
F FORTRAN
M ASSEMBLY LANGUAGE
P PL/1
PF PL/1 - FORMAC

ACM INDEX

F2 EIGENVALUES AND EIGENVECTORS OF MATRICES

F2 EIGENSYSTEM PACKAGE, EISPACK(F) AL,2-72,6-72
F2 SYMMETRIC-BISECTION, INV. ITN. BT,4-64(124)
F2 COMPUTING SU(N) INVARIANTS BT,11-71(1)
F2 SYMMETRIC TRIDIAGONAL MATRIX BT,11-71(262)
F2 85 JACOBI METHOD CA,5-62(208),5-62(440),
F2 85 6-63(447)
F2 104 REDUCTION-BAND TO TRIDIAGONAL CA,5-62(387)
F2 122 GIVENS TRIDIAGONAL REDUCTION CA,5-62(482),7-64(144)
F2 183 REDUCTION-BAND TO TRIDIAGONAL CA,6-63(315)
F2 253 SYMMETRIC QR-EIGENVALUES CA,8-65(217),10-67(376)
F2 254 SYMMETRIC QR-EIVALUES,EIVECTORS CA,8-65(218),10-67(376)
F2 270 EIGENVECTORS BY GAUSSIAN ELIM. CA,8-65(668)
F2 297 SYM.SYS.(A-LAM=B)X,EIVALS-VECS. CA,10-67(181)
F2 343 REAL GENERAL MATRIX CA,11-68(820),13-70(122
F2 343 13-70(694),15-72(466)
F2 384 EIVAL-EIVEC OF A SYMM. MATRIX CA,13-70(369),13-70(750
F2 405 ROOTS OF MATRIX PENCILS CA,14-71(113),15-72(107
F2 464 REAL, SYMM, TRIDIAGONAL MAT.(A) CA,16-73(689)
F2 HERMITIAN MAT EIGENPROBLEM H*X=E*S*X (F) CC,8-74(85)
F2 BANDED EIGENVALUE PROBLEMS (F) CC,10-75(30)
F2 TRIDIAGONAL SIMIL.BY ELIM. CJ,4-61(175)
F2 EIGENVALUES BY QR-ALGORITHM CJ,4-62(344)
F2 SYMM MAT-LLT AND STURM SEQ. CJ,9-66(303)
F2 32 EIVALS-REAL SYM MAT-DBL QR STP CJ,11-68(112)
F2 LR ALGRTHM FOR SYMM TRID MAT(A) CJ,15-72(268)
F2 JK METHOD FOR REAL SYMM MAT (F) CJ,15-72(271)
F2 EIGENVALUES OF QUINDIAGONAL MATRIX (A) CJ,18-75(70)
F2 93 GENERALIZED EIGENVALUE PROBLEM (F) CJ,20-77(86)
F2 EIGENVALUES-LAGUERRES METHOD MC,18-64(474),20-66(437
F2 EIVECS FOR GENL COMPLEX MAT MC,22-68(785)
F2 EIGENVALUES OF TRIDIAG. MATRIX NM,4-62(354)
F2 EIGENVECTORS OF TRIDIAG. MTRX NM,4-62(354)
F2 HOUSEHOLDERS METHOD NM,4-62(354)
F2 LR TRANSFORMATION METHOD NM,5-63(273)
F2 HOUSEHOLDER RED.-COMPLEX MAT. NM,8-66(79)
F2 JACOBI METHOD NM,9-66(
F2 EIGENVECTORS OF BAND MATRICES NM,9-67(285)
F2 EIVALUES OF SYMM.TRIDIAG.MATRIX NM,9-67(388)
F2 EIVALUES-EIVECTORS OF REAL MTRX NM,11-68(3)
F2 SYM EIPROBLEM A*X=LAM*B*X NM,11-68(102)
F2 RATIONAL QR FOR SYM TRIDIAG NM,11-68(268)
F2 MOD.LR.-CMPLX HESSENBERG MATRIX NM,12-68(372)
F2 IMPLICIT QL ALGORITHM NM,12-68(379)
F2 BALANCING A MATRIX NM,13-69(293)
F2 QR FOR REAL HESSENBERG MTRX NM,14-70(219)
F2 EIGENVEC OF REAL AND COMP MAT NM,16-70(181)
F2 SKEW-SYMMETRIC MATRIX NM,17-71(189)
F2 JACOBI TYPE METHOD FOR COMPLEX SYM. (A) NM,25-76(347)
F2 496 LZ ALG. FOR GEN. COMPLEX EIGENVALUES (F) TO,1-75(271),2-76(396)
F2 506 EIGENVALUES OF HESSENBERG MATRIX (F) TO,2-76(275)
F2 517 COND. NOS. OF EIGENVALUES (F) TO,3-77(186)
F2 530 SKEW-SYMMETRIC MATRICES (F) TO,4-78(286)
F2 535 GEN. EIGENPROBLEM FOR COMPLEX MAT. (F) TO,4-78( )
F2 538 EIGENSYSTEMS OF GEN. SYMM. MATRICES (F) TO,5-79(118)
F2 560 JORDAN NORMAL FORM OF COMPLEX MATRIX (F) TO,6-80(437)
F2 19 EIGENVALUES OF REAL MATRIX (A) ZM,13-72(283)
F2 225(37)NO.18
F2 225(37)NO.21

F3 DETERMINANTS

F3 82 DETERMINANT OF ORTHOGONAL MATRIX (F) AS,24-75(150)
F3 41 DETERMINANT EVALUATION CA,4-61(176),6-63(520),
F3 41 7-64(144),9-66(686)
F3 159 DETERMINANT EVALUATION CA,6-63(104),6-63(739)
F3 170 DETERMINANT-POLYNOMIAL ELEMENTS CA,6-63(165),6-63(450),
F3 170 7-64(421)
F3 224 EVALUATION OF DETERMINANT CA,7-64(243),7-64(702)
F3 269 DETERMINANT BY GAUSSIAN ELIM. CA,8-65(668),9-66(686)

F4 SIMULTANEOUS LINEAR EQUATIONS

F4 108 MULT. LINEAR REGRESSION IN L1 SENSE (F) AS,26-77(106),27-78(378
F4 108 MULT. LINEAR REGRESSION IN L1 SENSE (F) AS,26-77(106)
F4 132 LEAST ABSOLUTE VALUE ESTIMATION (F) AS,27-78(363)
F4 4 GAUSSIAN ELIMINATION BT,2-62(256),3-63(60)
F4 7 LINEAR SYSTEM WITH BAND MATRIX BT,3-63(207)
F4 12 GAUSSIAN ELIMINATION BT,5-65(64)
F4 68 IT REFINEMENT-LEAST SQR SOLN BT,8-68(20)
F4 27 LIN EQ WITH SYM NON-DEF MATRIX BT,10-70(386)
F4 16 CROUT WITH PIVOTING CA,3-60(507),3-60(540),
F4 16 4-61(154)
F4 17 SOLVE TRIDIAGONAL MATRIX CA,3-60(508)
F4 24 SOLVE TRIDIAGONAL MATRIX CA,3-60(602)
F4 43 CROUT WITH PIVOTING CA,4-61(176),4-61(182),
F4 43 6-63(445)
F4 92 SIMULT.EQNS.-ITERATIVE SOLN. CA,5-62(286)
F4 107 GAUSSIAN ELIMINATION CA,5-62(388),6-63(39),
F4 107 6-63(445)
F4 126 GAUSSIAN ELIMINATION CA,5-62(511)
F4 135 CROUT WITH EQUILIBRATION CA,5-62(553),5-62(557),
F4 135 7-64(421),8-65(104)
F4 195 BAND SOLVE CA,6-63(441),15-72(1074
F4 220 GAUSS+SEIDEL CA,6-63(739),7-64(349)
F4 238 CONJUGATE GRADIENT METHOD CA,7-64(481)
F4 288 LINEAR DIOPHANTINE EQUATIONS CA,9-66(514)
F4 290 EXACT SOLUTION OF LINEAR EQNS. CA,9-66(683)
F4 328 CHEBY SOLN-OVERDET LINEAR SYS CA,11-68(428),12-69(326
F4 358 SING.VAL.DECOMP.-COMPLEX MATRIX CA,12-69(564)
F4 406 SOLUTION BY RESIDUE ARITH. (F) CA,14-71(180),16-73(311
F4 408 SPARSE MAT. PACKAGE(PART 1) (F) CA,14-71(265),16-73(311
F4 408 16-73(578)
F4 408 A SPARSE MATRIX PACKAGE (PART 1) (F) CA,14-71(265),16-73(311
F4 408 A SPARSE MATRIX PACKAGE (F) CA,14-71(265),16-73(311
F4 408 16-73(578),3-77(303)
F4 16-73(578),3-77(303)
F4 4-78(295),6-80(456)
F4 408 SPARSE MAT. PACKAGE (PART 1) (F) CA,14-71(265),16-73(311
F4 16-73(578),3-77(303),
F4 4-78(295)
F4 423 LINEAR EQUATION SOLVER CA,15-72(274)
F4 432 SOLUTION TO EQUATION AX+XB=C CA,15-72(820)
F4 449 SOL LIN PROG PROB 0-1 VAR (F) CA,16-73(445)
F4 470 ALMOST TRIDIAGONAL MATRIX (F) CA,16-73(760)
F4 478 OVERDETERMINED SYSTEM IN L1 NORM (F) CA,17-74(319),18-75(277
F4 DETERMINATION OF SSOR-SI ITER. PARAM.(F) CC,10-75(194)
F4 SOLUTION WITH REL ERR ESTIMATE CJ,11-68(92)
F4 51 GENL. 3-TERM LINEAR SYSTEMS CJ,13-70(324)
F4 69 SYMMETRIC THREE TERM SYSTEMS CJ,14-71(444)
F4 TRIDIAGONAL SYSTEMS (A) CJ,15-72(356)
F4 80 QUINDIAGONAL SYSTEMS (A) CJ,16-73(278)
F4 82 PARTIAL PIVOTING, REDUCED STORAGE (A) CJ,17-74(377)
F4 84 TRIDIAGONAL SYSTEMS COUPLED ALGORITHM(A) CJ,17-74(378)
F4 84 COUPLED ALG. FOR TRIDIAGONAL SYSTEMS (A) CJ,17-74(378),20-77(91)
F4 SOL. OF CERTAIN BANDED LINEAR SYSTEMS(F) CJ,19-76(184)
F4 BANDED LINEAR SYSTEMS (F) CJ,19-76(184)
F4 LARGE SPARSE LINEAR EQUATIONS (P) CJ,19-76(352)
F4 SOLUTION OF TRIDIAGONAL LINEAR SYS. (A) CJ,20-77(181)
F4 96 LIN. EQNS. FROM GALERKIN METHOD (A) CJ,20-77(371)
F4 98 TRIDIAGONAL SYSTEMS OF EQUATIONS (A) CJ,20-77(376)
F4 108 TRIDAIGONAL LINEAR SYSTEMS (A) CJ,22-79(283)
F4 ERROR BOUNDS FOR SYS LIN EQU CP,6-70(28)
F4 ANTISYMMETRICAL NETWORKS (A) CP,11-73(45)
F4 LIN CONSTRAINED OPTIM. PROBS(F) HA,AERE-R6354
F4 SOL LARGE SPARSE SETS LIN EQ(F) HA,AERE-R6545
F4 SOL OF SPARSE SETS LIN EQ (F) HA,AERE-R6844
F4 SOL OF LINEAR EQUATIONS (F) HA,AERE-R6899
F4 SPARSE, SYMM, & POS-DEFINITE(F) HA,AERE-R7119
F4 TRIDIAGONAL SYSTEMS (A) JC,20-73(27)
F4 VANDERMONDE SYSTEM OF EQUATION MC,24-70(893)
F4 CONJUGATE GRADIENT METHOD NM,5-63(195)
F4 LEAST SQUARES SOLUTION NM,7-65(271)
F4 ELIM.WITH WEIGHTED ROW COMB. NM,7-65(341)
F4 ITER.REFIN.-SOLN.OF POS.DEF.MTX NM,8-66(206)
F4 REAL AND COMPLEX LINEAR SYSTEM NM,8-66(222)
F4 SYMM.AND UNSYMM.BAND EQUATIONS NM,9-67(285)
F4 VANDERMONDE SYSTEMS NM,18-71(44)
F4 DECOMPOSITION OF A SYMMETRIC MATRIX (A) NM,27-76(95)
F4 495 OVERDETERMINED SYST IN CHEBYSHEV NORM(F) TO,1-75(264)
F4 512 POS. DEF. SYM. QUINDIAGONAL LIN. (F) TO,3-77(96)
F4 522 EXACT SOL. OF INTEGER LINEAR SYSTEMS (F) TO,3-77(404)
F4 533 SPARSE GAUSSIAN ELIMINATION (F) TO,4-78( )
F4 544 WEIGHTED LEAST SQ. ITER. REFINEMENT (F) TO,5-79( )
F4 546 ALMOST BLOCK DIAGONAL SYSTEMS (F) TO,6-80(88)
F4 551 L1 SOLUTION OF OVERDETERMINED SYSTEMS(F) TO,6-80(228)
F4 552 CONSTRAINED L1 LINEAR APPROXIMATION (F) TO,6-80(231)
F4 563 LINEAR CONSTRAINED DISCR. L1 APPROX. (F) TO,6-80( )
F4 564 TEST PROBLEM GEN. FOR L1 APPROX. (F) TO,6-80( )
F4 29 SOKOLOV'S METHOD (A) ZM,14-74(127)
F4 37 SOL. OF SPARSE LINEAR EQ. SYSTEMS (A) ZM,14-74(481)

F5 ORTHOGONALIZATION

F5 10 USE OF ORTHOGONAL POLYNOMIALS AS,17-68(283),20-71(118
F5 10 20-71(216)
F5 42 ORTHOGONAL POLYNOMIALS AS,20-71(209)
F5 46 GRAM-SCHMIDT ORTHOGONALIZATION AS,20-71(335)
F5 127 ORTHONORMALIZATION CA,5-62(511),13-70(122)
F5 358 SING.VAL.DECOMP.-COMPLEX MATRIX CA,12-69(564)
F5 SCHMIDT ORTHONORMALIZATION CP,1-66(159)

G1 SIMPLE CALCULATIONS ON STATISTICAL DATA

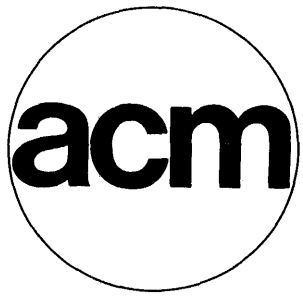
G1 16 ESTIMATION FROM NORMAL DATA AS,18-69(110)
G1 16 ESTIMATION FROM NORMAL DATA AS,18-69(110),26-77(122
G1 19 VARIANCE OF FACTORIAL TABLE AS,18-69(199)
G1 22 THE INTERACTION ALGORITHM AS,18-69(283)
G1 23 CALCULATION OF EFFECTS AS,18-69(287)
G1 25 MEANS FROM VARIANCE ANALYSIS AS,18-69(294)
G1 29 THE RUNS UP AND DOWN TEST AS,19-70(190),25-76(193
G1 31 BINOMIAL SEQUENTIAL SAMPLING AS,19-70(197)
G1 33 HYPERGEOMETRIC SAMPLE SIZES AS,19-70(287)
G1 35 FINITE POPULATION PROBAB. (A) AS,20-71(99),20-71(346)
G1 35 21-72(352)
G1 35 PROB. FROM FINITE POPULATIONS (A) AS,20-71(99),20-71(346)
G1 36 EXACT CONFIDENCE LIMITS AS,20-71(105)
G1 41 MATRIX MEAN AND DISPERSION AS,20-71(206)
G1 48 UNCERTAINTY FUN. FOR BINARY SEQ AS,21-72(97)
G1 53 WISHART VARIATE GENERATOR (F) AS,21-72(341)
G1 55 MANN-WHITNEY U-STATISTIC (P) AS,21-72(348)
G1 59 HYPERGEOMETRIC PROBABILITIES(F) AS,22-73(130)
G1 62 MANN-WHITNEY U-STATISTIC (F) AS,22-73(269)
G1 67 ABSORPTION IN SEQ BINOM SAMP(F) AS,23-74(83)
G1 68 TRUNCATED NEG. BIN. DISTR. (F) AS,23-74(87)
G1 70 PERCENTAGE PTS OF NORM DIST (F) AS,23-74(96)
G1 72 MEAN VECTORS & DISPERSION MATRICES (F) AS,23-74(234)
G1 77 NULL DIST OF LARGEST ROOT STATISTIC (F) AS,23-74(458)
G1 78 THE MEDIANCENTRE (F) AS,23-74(466),24-75(390
G1 80 SPHERICAL STATISTICS (A) AS,24-75(144)
G1 81 CIRCULAR STATISTICS (A) AS,24-75(147)
G1 84 MULTIVARIATE SKEWNESS AND KURTOSIS (F) AS,24-75(262)

Collected Algorithms from ACM

Volume III

Collected Algorithms from ACM

Volume III
Algorithms 493 ff.



1981

ACM Algorithms 493ff. are available by purchase in the form of listing, or card deck, or magnetic tape (9 track EBCDIC, 9 track ASCII, 7 track BCD), from ACM Algorithms Distribution Service, c/o International Mathematical & Statistical Libraries, Inc., Sixth Floor, GNB Building, 7500 Bellaire Boulevard, Houston, TX 77036.

ALGORITHM 493
Zeros of a Real Polynomial [C2]

M.A. JENKINS

Queen's University

Key Words and Phrases: roots, zeros of a polynomial
CR Categories: 5.15
Language: Fortran

DESCRIPTION

The subroutine RPOLY is a Fortran program to find all the zeros of a real polynomial. The parameters are:

OP	double precision vector of coefficients in order of decreasing powers of the variable
DEGREE	integer degree of the polynomial
ZEROR, ZEROI	double precision vectors of real and imaginary parts of the zeros found by the algorithm
FAIL	logical parameter which is true only if the leading coefficient is zero or if RPOLY has found fewer than degree zeros; in the latter case the degree is reset to the number of zeros found.

The routine as written solves polynomials of degree up to 100; however, this can be modified by systematic changing of the declarations in the routines.

The program is based on the three-stage algorithm described in Jenkins and Traub [1]. The algorithm generates a sequence of polynomials of degree one less than the degree of the given polynomial from which an approximation to a zero or a quadratic factor can be extracted. The first stage is linearly convergent and involves no shift of origin. It is used primarily to bias the decision making process in the second stage in favor of the zeros of small magnitude. The second stage is also linearly convergent and involves a double shift to a complex point and its conjugate. The shift point is chosen arbitrarily on a circle whose radius is less than the magnitude of all the zeros. In most cases, either the shift is closest to a real zero, or the pair of shift points are equidistant and closest to a pair of zeros. In the former case the second stage yields an approximation to the real zero and in the latter case it yields an approximation to the real quadratic factor. The third stage involves one of two variable-shift iterations, where the latest approximation(s) is used as the shift point(s) for the generation of the next polynomial. The choice of iteration is based on which case is observed in stage two. The convergence is superquadratic and usually requires only a few steps. The decision processes are made in a fail-safe manner. If the third stage is entered prematurely or the incorrect iteration is chosen, the second stage is resumed. If no convergence is observed in the second stage after a fixed number of steps, a new shift is chosen and the second stage is restarted with a higher fixed limit. The third-stage iterations are terminated when a stopping

Received 11 Feb. 1974 and 16 Oct. 1974.
Author's address: Department of Computing and Information Science, Queen's University, Kingston, Ont., Canada K7L 3N6.

criterion based on roundoff error analysis has been satisfied. The real zero or quadratic factor is removed by polynomial deflation and the algorithm is repeated on the reduced polynomial.

The first statements of RPOLY set the following four constants which describe the floating-point arithmetic of the computer being used:

ETA	maximum relative representation error, which can be described as the smallest positive floating-point number such that 1 + ETA > 1 in floating-point arithmetic
INFIN	large floating-point number near the top of the range
SMALNO	small positive floating-point number near zero
BASE	exponent base for the floating-point number system.

The program is written in a portable subset of standard Fortran. It has been successfully used on the Burroughs B6700 and the IBM 360/50.

The program has been tested on a large number of polynomials, some chosen to test weaknesses common to zerofinding routines, others randomly generated by a number of techniques.

REFERENCES

1. JENKINS, M.A., AND TRAUB, J.F. A three-stage algorithm for real polynomials using quadratic iteration. *SIAM J. Numer. Anal. 7* (1970), 545–566.
2. JENKINS, M.A., AND TRAUB, J.F. Principles for testing polynomial zerofinding programs. *ACM TOMS 1*, 1 (March 1975), 26–34.

ALGORITHM

```
      SUBROUTINE RPOLY(OP, DEGREE, ZEROR, ZEROI,                        RPO   10
     * FAIL)                                                            RPO   20
C FINDS THE ZEROS OF A REAL POLYNOMIAL                                  RPO   30
C OP  - DOUBLE PRECISION VECTOR OF COEFFICIENTS IN                      RPO   40
C       ORDER OF DECREASING POWERS.                                     RPO   50
C DEGREE   - INTEGER DEGREE OF POLYNOMIAL.                              RPO   60
C ZEROR, ZEROI - OUTPUT DOUBLE PRECISION VECTORS OF                     RPO   70
C                REAL AND IMAGINARY PARTS OF THE                        RPO   80
C                ZEROS.                                                 RPO   90
C FAIL  - OUTPUT LOGICAL PARAMETER, TRUE ONLY IF                        RPO  100
C         LEADING COEFFICIENT IS ZERO OR IF RPOLY                       RPO  110
C         HAS FOUND FEWER THAN DEGREE ZEROS.                            RPO  120
C         IN THE LATTER CASE DEGREE IS RESET TO                         RPO  130
C         THE NUMBER OF ZEROS FOUND.                                    RPO  140
C TO CHANGE THE SIZE OF POLYNOMIALS WHICH CAN BE                        RPO  150
C SOLVED, RESET THE DIMENSIONS OF THE ARRAYS IN THE                     RPO  160
C COMMON AREA AND IN THE FOLLOWING DECLARATIONS.                        RPO  170
C THE SUBROUTINE USES SINGLE PRECISION CALCULATIONS                     RPO  180
C FOR SCALING, BOUNDS AND ERROR CALCULATIONS. ALL                       RPO  190
C CALCULATIONS FOR THE ITERATIONS ARE DONE IN DOUBLE                    RPO  200
C PRECISION.                                                            RPO  210
      COMMON /GLOBAL/ P, QP, K, QK, SVK, SR, SI, U,                     RPO  220
     * V, A, B, C, D, A1, A2, A3, A6, A7, E, F, G,                      RPO  230
     * H, SZR, SZI, LZR, LZI, ETA, ARE, MRE, N, NN                      RPO  240
      DOUBLE PRECISION P(101), QP(101), K(101),                         RPO  250
     * QK(101), SVK(101), SR, SI, U, V, A, B, C, D,                     RPO  260
     * A1, A2, A3, A6, A7, E, F, G, H, SZR, SZI,                        RPO  270
     * LZR, LZI                                                         RPO  280
      REAL ETA, ARE, MRE                                                RPO  290
      INTEGER N, NN                                                     RPO  300
      DOUBLE PRECISION OP(101), TEMP(101),                              RPO  310
     * ZEROR(100), ZEROI(100), T, AA, BB, CC, DABS,                     RPO  320
     * FACTOR                                                           RPO  330
      REAL PT(101), LO, MAX, MIN, XX, YY, COSR,                         RPO  340
     * SINR, XXX, X, SC, BND, XM, FF, DF, DX, INFIN,                    RPO  350
     * SMALNO, BASE                                                     RPO  360
      INTEGER DEGREE, CNT, NZ, I, J, JJ, NM1                            RPO  370
      LOGICAL FAIL, ZEROK                                               RPO  380
C THE FOLLOWING STATEMENTS SET MACHINE CONSTANTS USED                   RPO  390
C IN VARIOUS PARTS OF THE PROGRAM. THE MEANING OF THE                   RPO  400
C FOUR CONSTANTS ARE...                                                 RPO  410
C ETA     THE MAXIMUM RELATIVE REPRESENTATION ERROR                     RPO  420
C         WHICH CAN BE DESCRIBED AS THE SMALLEST                        RPO  430
C         POSITIVE FLOATING POINT NUMBER SUCH THAT                      RPO  440
C         1.D0+ETA IS GREATER THAN 1.                                   RPO  450
C INFINY  THE LARGEST FLOATING-POINT NUMBER.                            RPO  460
C SMALNO  THE SMALLEST POSITIVE FLOATING-POINT NUMBER                   RPO  470
C         IF THE EXPONENT RANGE DIFFERS IN SINGLE AND                   RPO  480
C         DOUBLE PRECISION THEN SMALNO AND INFIN                        RPO  490
C         SHOULD INDICATE THE SMALLER RANGE.                            RPO  500
C BASE    THE BASE OF THE FLOATING-POINT NUMBER                         RPO  510
```

```
C         SYSTEM USED.                                                  RPO  520
C THE VALUES BELOW CORRESPOND TO THE BURROUGHS B6700                    RPO  530
      BASE = 8.                                                         RPO  540
      ETA = .5*BASE**(1-26)                                             RPO  550
      INFIN = 4.3E68                                                    RPO  560
      SMALNO = 1.0E-45                                                  RPO  570
C ARE AND MRE REFER TO THE UNIT ERROR IN + AND *                        RPO  580
C RESPECTIVELY. THEY ARE ASSUMED TO BE THE SAME AS                      RPO  590
C ETA.                                                                  RPO  600
      ARE = ETA                                                         RPO  610
      MRE = ETA                                                         RPO  620
      LO = SMALNO/ETA                                                   RPO  630
C INITIALIZATION OF CONSTANTS FOR SHIFT ROTATION                        RPO  640
      XX = .70710678                                                    RPO  650
      YY = -XX                                                          RPO  660
      COSR = -.069756474                                                RPO  670
      SINR = .99756405                                                  RPO  680
      FAIL = .FALSE.                                                    RPO  690
      N = DEGREE                                                        RPO  700
      NN = N + 1                                                        RPO  710
C ALGORITHM FAILS IF THE LEADING COEFFICIENT IS ZERO.                   RPO  720
      IF (OP(1).NE.0.D0) GO TO 10                                       RPO  730
      FAIL = .TRUE.                                                     RPO  740
      DEGREE = 0                                                        RPO  750
      RETURN                                                            RPO  760
C REMOVE THE ZEROS AT THE ORIGIN IF ANY                                 RPO  770
   10 IF (OP(NN).NE.0.0D0) GO TO 20                                     RPO  780
      J = DEGREE - N + 1                                                RPO  790
      ZEROR(J) = 0.D0                                                   RPO  800
      ZEROI(J) = 0.D0                                                   RPO  810
      NN = NN - 1                                                       RPO  820
      N = N - 1                                                         RPO  830
      GO TO 10                                                          RPO  840
C MAKE A COPY OF THE COEFFICIENTS                                       RPO  850
   20 DO 30 I=1,NN                                                      RPO  860
        P(I) = OP(I)                                                    RPO  870
   30 CONTINUE                                                          RPO  880
C START THE ALGORITHM FOR ONE ZERO                                      RPO  890
   40 IF (N.GT.2) GO TO 60                                              RPO  900
      IF (N.LT.1) RETURN                                                RPO  910
C CALCULATE THE FINAL ZERO OR PAIR OF ZEROS                             RPO  920
      IF (N.EQ.2) GO TO 50                                              RPO  930
      ZEROR(DEGREE) = -P(2)/P(1)                                        RPO  940
      ZEROI(DEGREE) = 0.0D0                                             RPO  950
      RETURN                                                            RPO  960
   50 CALL QUAD(P(1), P(2), P(3), ZEROR(DEGREE-1),                      RPO  970
     * ZEROI(DEGREE-1), ZEROR(DEGREE), ZEROI(DEGREE))                   RPO  980
      RETURN                                                            RPO  990
C FIND LARGEST AND SMALLEST MODULI OF COEFFICIENTS.                     RPO 1000
   60 MAX = 0.                                                          RPO 1010
      MIN = INFIN                                                       RPO 1020
      DO 70 I=1,NN                                                      RPO 1030
        X = ABS(SNGL(P(I)))                                             RPO 1040
        IF (X.GT.MAX) MAX = X                                           RPO 1050
        IF (X.NE.0. .AND. X.LT.MIN) MIN = X                             RPO 1060
   70 CONTINUE                                                          RPO 1070
C SCALE IF THERE ARE LARGE OR VERY SMALL COEFFICIENTS                   RPO 1080
C COMPUTES A SCALE FACTOR TO MULTIPLY THE                               RPO 1090
C COEFFICIENTS OF THE POLYNOMIAL. THE SCALING IS DONE                   RPO 1100
C TO AVOID OVERFLOW AND TO AVOID UNDETECTED UNDERFLOW                   RPO 1110
C INTERFERING WITH THE CONVERGENCE CRITERION.                           RPO 1120
C THE FACTOR IS A POWER OF THE BASE                                     RPO 1130
      SC = LO/MIN                                                       RPO 1140
      IF (SC.GT.1.0) GO TO 80                                           RPO 1150
      IF (MAX.LT.10.) GO TO 110                                         RPO 1160
      IF (SC.EQ.0.) SC = SMALNO                                         RPO 1170
      GO TO 90                                                          RPO 1180
   80 IF (INFIN/SC.LT.MAX) GO TO 110                                    RPO 1190
   90 L = ALOG(SC)/ALOG(BASE) + .5                                      RPO 1200
      FACTOR = (BASE*1.0D0)**L                                          RPO 1210
      IF (FACTOR.EQ.1.D0) GO TO 110                                     RPO 1220
      DO 100 I=1,NN                                                     RPO 1230
        P(I) = FACTOR*P(I)                                              RPO 1240
  100 CONTINUE                                                          RPO 1250
C COMPUTE LOWER BOUND ON MODULI OF ZEROS.                               RPO 1260
  110 DO 120 I=1,NN                                                     RPO 1270
        PT(I) = ABS(SNGL(P(I)))                                         RPO 1280
  120 CONTINUE                                                          RPO 1290
      PT(NN) = -PT(NN)                                                  RPO 1300
C COMPUTE UPPER ESTIMATE OF BOUND                                       RPO 1310
      X = EXP((ALOG(-PT(NN))-ALOG(PT(1)))/FLOAT(N))                     RPO 1320
      IF (PT(N).EQ.0.) GO TO 130                                        RPO 1330
C IF NEWTON STEP AT THE ORIGIN IS BETTER, USE IT.                       RPO 1340
      XM = -PT(NN)/PT(N)                                                RPO 1350
      IF (XM.LT.X) X = XM                                               RPO 1360
C CHOP THE INTERVAL (0,X) UNTIL FF .LE. 0                               RPO 1370
  130 XM = X*.1                                                         RPO 1380
      FF = PT(1)                                                        RPO 1390
      DO 140 I=2,NN                                                     RPO 1400
        FF = FF*XM + PT(I)                                              RPO 1410
  140 CONTINUE                                                          RPO 1420
      IF (FF.LE.0.) GO TO 150                                           RPO 1430
```

```
      X = XM                                                        RPO 1440
      GO TO 130                                                     RPO 1450
  150 DX = X                                                        RPO 1460
C DO NEWTON ITERATION UNTIL X CONVERGES TO TWO                      RPO 1470
C DECIMAL PLACES                                                    RPO 1480
  160 IF (ABS(DX/X).LE..005) GO TO 180                              RPO 1490
      FF = PT(1)                                                    RPO 1500
      DF = FF                                                       RPO 1510
      DO 170 I=2,N                                                  RPO 1520
        FF = FF*X + PT(I)                                           RPO 1530
        DF = DF*X + FF                                              RPO 1540
  170 CONTINUE                                                      RPO 1550
      FF = FF*X + PT(NN)                                            RPO 1560
      DX = FF/DF                                                    RPO 1570
      X = X - DX                                                    RPO 1580
      GO TO 160                                                     RPO 1590
  180 BND = X                                                       RPO 1600
C COMPUTE THE DERIVATIVE AS THE INTIAL K POLYNOMIAL                 RPO 1610
C AND DO 5 STEPS WITH NO SHIFT                                      RPO 1620
      NM1 = N - 1                                                   RPO 1630
      DO 190 I=2,N                                                  RPO 1640
        K(I) = FLOAT(NN-I)*P(I)/FLOAT(N)                            RPO 1650
  190 CONTINUE                                                      RPO 1660
      K(1) = P(1)                                                   RPO 1670
      AA = P(NN)                                                    RPO 1680
      BB = P(N)                                                     RPO 1690
      ZEROK = K(N).EQ.0.D0                                          RPO 1700
      DO 230 JJ=1,5                                                 RPO 1710
        CC = K(N)                                                   RPO 1720
        IF (ZEROK) GO TO 210                                        RPO 1730
C USE SCALED FORM OF RECURRENCE IF VALUE OF K AT 0 IS               RPO 1740
C NONZERO                                                           RPO 1750
        T = -AA/CC                                                  RPO 1760
        DO 200 I=1,NM1                                              RPO 1770
          J = NN - I                                                RPO 1780
          K(J) = T*K(J-1) + P(J)                                    RPO 1790
  200   CONTINUE                                                    RPO 1800
        K(1) = P(1)                                                 RPO 1810
        ZEROK = DABS(K(N)).LE.DABS(BB)*ETA*10.                      RPO 1820
        GO TO 230                                                   RPO 1830
C USE UNSCALED FORM OF RECURRENCE                                   RPO 1840
  210   DO 220 I=1,NM1                                              RPO 1850
          J = NN - I                                                RPO 1860
          K(J) = K(J-1)                                             RPO 1870
  220   CONTINUE                                                    RPO 1880
        K(1) = 0.D0                                                 RPO 1890
        ZEROK = K(N).EQ.0.D0                                        RPO 1900
  230 CONTINUE                                                      RPO 1910
C SAVE K FOR RESTARTS WITH NEW SHIFTS                               RPO 1920
      DO 240 I=1,N                                                  RPO 1930
        TEMP(I) = K(I)                                              RPO 1940
  240 CONTINUE                                                      RPO 1950
C LOOP TO SELECT THE QUADRATIC  CORRESPONDING TO EACH               RPO 1960
C NEW SHIFT                                                         RPO 1970
      DO 280 CNT=1,20                                               RPO 1980
C QUADRATIC CORRESPONDS TO A DOUBLE SHIFT TO A                      RPO 1990
C NON-REAL POINT AND ITS COMPLEX CONJUGATE. THE POINT               RPO 2000
C HAS MODULUS BND AND AMPLITUDE ROTATED BY 94 DEGREES               RPO 2010
C FROM THE PREVIOUS SHIFT                                           RPO 2020
        XXX = COSR*XX - SINR*YY                                     RPO 2030
        YY = SINR*XX + COSR*YY                                      RPO 2040
        XX = XXX                                                    RPO 2050
        SR = BND*XX                                                 RPO 2060
        SI = BND*YY                                                 RPO 2070
        U = -2.0D0*SR                                               RPO 2080
        V = BND                                                     RPO 2090
C SECOND STAGE CALCULATION, FIXED QUADRATIC                         RPO 2100
        CALL FXSHFR(20*CNT, NZ)                                     RPO 2110
        IF (NZ.EQ.0) GO TO 260                                      RPO 2120
C THE SECOND STAGE JUMPS DIRECTLY TO ONE OF THE THIRD               RPO 2130
C STAGE ITERATIONS AND RETURNS HERE IF SUCCESSFUL.                  RPO 2140
C DEFLATE THE POLYNOMIAL, STORE THE ZERO OR ZEROS AND               RPO 2150
C RETURN TO THE MAIN ALGORITHM.                                     RPO 2160
        J = DEGREE - N + 1                                          RPO 2170
        ZEROR(J) = SZR                                              RPO 2180
        ZEROI(J) = SZI                                              RPO 2190
        NN = NN - NZ                                                RPO 2200
        N = NN - 1                                                  RPO 2210
        DO 250 I=1,NN                                               RPO 2220
          P(I) = QP(I)                                              RPO 2230
  250   CONTINUE                                                    RPO 2240
        IF (NZ.EQ.1) GO TO 40                                       RPO 2250
        ZEROR(J+1) = LZR                                            RPO 2260
        ZEROI(J+1) = LZI                                            RPO 2270
        GO TO 40                                                    RPO 2280
C IF THE ITERATION IS UNSUCCESSFUL ANOTHER QUADRATIC                RPO 2290
C IS CHOSEN AFTER RESTORING K                                       RPO 2300
  260   DO 270 I=1,N                                                RPO 2310
          K(I) = TEMP(I)                                            RPO 2320
  270   CONTINUE                                                    RPO 2330
  280 CONTINUE                                                      RPO 2340
C RETURN WITH FAILURE IF NO CONVERGENCE WITH 20                     RPO 2350
```

```
C SHIFTS                                                          RPO 2360
      FAIL = .TRUE.                                               RPO 2370
      DEGREE = DEGREE - N                                         RPO 2380
      RETURN                                                      RPO 2390
      END                                                         RPO 2400



      SUBROUTINE FXSHFR(L2, NZ)                                   FXS   10
C COMPUTES UP TO  L2  FIXED SHIFT K-POLYNOMIALS,                  FXS   20
C TESTING FOR CONVERGENCE IN THE LINEAR OR QUADRATIC              FXS   30
C CASE. INITIATES ONE OF THE VARIABLE SHIFT                       FXS   40
C ITERATIONS AND RETURNS WITH THE NUMBER OF ZEROS                 FXS   50
C FOUND.                                                          FXS   60
C L2 - LIMIT OF FIXED SHIFT STEPS                                 FXS   70
C NZ - NUMBER OF ZEROS FOUND                                      FXS   80
      COMMON /GLOBAL/ P, QP, K, QK, SVK, SR, SI, U,               FXS   90
     * V, A, B, C, D, A1, A2, A3, A6, A7, E, F, G,                FXS  100
     * H, SZR, SZI, LZR, LZI, ETA, ARE, MRE, N, NN                FXS  110
      DOUBLE PRECISION P(101), QP(101), K(101),                   FXS  120
     * QK(101), SVK(101), SR, SI, U, V, A, B, C, D,               FXS  130
     * A1, A2, A3, A6, A7, E, F, G, H, SZR, SZI,                  FXS  140
     * LZR, LZI                                                   FXS  150
      REAL ETA, ARE, MRE                                          FXS  160
      INTEGER N, NN                                               FXS  170
      DOUBLE PRECISION SVU, SVV, UI, VI, S                        FXS  180
      REAL BETAS, BETAV, OSS, OVV, SS, VV, TS, TV,                FXS  190
     * OTS. OTV, TVV, TSS                                         FXS  200
      INTEGER L2, NZ, TYPE, I, J, IFLAG                           FXS  210
      LOGICAL VPASS, SPASS, VTRY, STRY                            FXS  220
      NZ = 0                                                      FXS  230
      BETAV = .25                                                 FXS  240
      BETAS = .25                                                 FXS  250
      OSS = SR                                                    FXS  260
      OVV = V                                                     FXS  270
C EVALUATE POLYNOMIAL BY SYNTHETIC DIVISION                       FXS  280
      CALL QUADSD(NN, U, V, P, QP, A, B)                          FXS  290
      CALL CALCSC(TYPE)                                           FXS  300
      DO 80 J=1,L2                                                FXS  310
C CALCULATE NEXT K POLYNOMIAL AND ESTIMATE V                      FXS  320
        CALL NEXTK(TYPE)                                          FXS  330
        CALL CALCSC(TYPE)                                         FXS  340
        CALL NEWEST(TYPE, UI, VI)                                 FXS  350
        VV = VI                                                   FXS  360
C ESTIMATE S                                                      FXS  370
        SS = 0.                                                   FXS  380
        IF (K(N).NE.0.D0) SS = -P(NN)/K(N)                        FXS  390
        TV = 1.                                                   FXS  400
        TS = 1.                                                   FXS  410
        IF (J.EQ.1 .OR. TYPE.EQ.3) GO TO 70                       FXS  420
C COMPUTE RELATIVE MEASURES OF CONVERGENCE OF S AND V             FXS  430
C SEQUENCES                                                       FXS  440
        IF (VV.NE.0.) TV = ABS((VV-OVV)/VV)                       FXS  450
        IF (SS.NE.0.) TS = ABS((SS-OSS)/SS)                       FXS  460
C IF DECREASING, MULTIPLY TWO MOST RECENT                         FXS  470
C CONVERGENCE MEASURES                                            FXS  480
        TVV = 1.                                                  FXS  490
        IF (TV.LT.OTV) TVV = TV*OTV                               FXS  500
        TSS = 1.                                                  FXS  510
        IF (TS.LT.OTS) TSS = TS*OTS                               FXS  520
C COMPARE WITH CONVERGENCE CRITERIA                               FXS  530
        VPASS = TVV.LT.BETAV                                      FXS  540
        SPASS = TSS.LT.BETAS                                      FXS  550
        IF (.NOT.(SPASS .OR. VPASS)) GO TO 70                     FXS  560
C AT LEAST ONE SEQUENCE HAS PASSED THE CONVERGENCE                FXS  570
C TEST. STORE VARIABLES BEFORE ITERATING                          FXS  580
        SVU = U                                                   FXS  590
        SVV = V                                                   FXS  600
        DO 10 I=1,N                                               FXS  610
          SVK(I) = K(I)                                           FXS  620
   10   CONTINUE                                                  FXS  630
        S = SS                                                    FXS  640
C CHOOSE ITERATION ACCORDING TO THE FASTEST                       FXS  650
C CONVERGING SEQUENCE                                             FXS  660
        VTRY = .FALSE.                                            FXS  670
        STRY = .FALSE.                                            FXS  680
        IF (SPASS .AND. ((.NOT.VPASS) .OR.                        FXS  690
     *   TSS.LT.TVV)) GO TO 40                                    FXS  700
   20   CALL QUADIT(UI, VI, NZ)                                   FXS  710
        IF (NZ.GT.0) RETURN                                       FXS  720
C QUADRATIC ITERATION HAS FAILED. FLAG THAT IT HAS                FXS  730
C BEEN TRIED AND DECREASE THE CONVERGENCE CRITERION.              FXS  740
        VTRY = .TRUE.                                             FXS  750
        BETAV = BETAV*.25                                         FXS  760
C TRY LINEAR ITERATION IF IT HAS NOT BEEN TRIED AND               FXS  770
C THE S SEQUENCE IS CONVERGING                                    FXS  780
        IF (STRY .OR. (.NOT.SPASS)) GO TO 50                      FXS  790
        DO 30 I=1,N                                               FXS  800
          K(I) = SVK(I)                                           FXS  810
   30   CONTINUE                                                  FXS  820
   40   CALL REALIT(S, NZ, IFLAG)                                 FXS  830
        IF (NZ.GT.0) RETURN                                       FXS  840
```

```
C LINEAR ITERATION HAS FAILED. FLAG THAT IT HAS BEEN                FXS  850
C TRIED AND DECREASE THE CONVERGENCE CRITERION                      FXS  860
        STRY = .TRUE.                                               FXS  870
        BETAS = BETAS*.25                                           FXS  880
        IF (IFLAG.EQ.0) GO TO 50                                    FXS  890
C IF LINEAR ITERATION SIGNALS AN ALMOST DOUBLE REAL                 FXS  900
C ZERO ATTEMPT QUADRATIC INTERATION                                 FXS  910
        UI = -(S+S)                                                 FXS  920
        VI = S*S                                                    FXS  930
        GO TO 20                                                    FXS  940
C RESTORE VARIABLES                                                 FXS  950
   50   U = SVU                                                     FXS  960
        V = SVV                                                     FXS  970
        DO 60 I=1,N                                                 FXS  980
          K(I) = SVK(I)                                             FXS  990
   60   CONTINUE                                                    FXS 1000
C TRY QUADRATIC ITERATION IF IT HAS NOT BEEN TRIED                  FXS 1010
C AND THE V SEQUENCE IS CONVERGING                                  FXS 1020
        IF (VPASS .AND. (.NOT.VTRY)) GO TO 20                       FXS 1030
C RECOMPUTE QP AND SCALAR VALUES TO CONTINUE THE                    FXS 1040
C SECOND STAGE                                                      FXS 1050
        CALL QUADSD(NN, U, V, P, QP, A, B)                          FXS 1060
        CALL CALCSC(TYPE)                                           FXS 1070
   70   OVV = VV                                                    FXS 1080
        OSS = SS                                                    FXS 1090
        OTV = TV                                                    FXS 1100
        OTS = TS                                                    FXS 1110
   80 CONTINUE                                                      FXS 1120
      RETURN                                                        FXS 1130
      END                                                           FXS 1140
```

```
      SUBROUTINE QUADIT(UU, VV, NZ)                                 QUA   10
C VARIABLE-SHIFT K-POLYNOMIAL ITERATION FOR A                       QUA   20
C QUADRATIC FACTOR CONVERGES ONLY IF THE ZEROS ARE                  QUA   30
C EQUIMODULAR OR NEARLY SO.                                         QUA   40
C UU,VV - COEFFICIENTS OF STARTING QUADRATIC                        QUA   50
C NZ - NUMBER OF ZERO FOUND                                         QUA   60
      COMMON /GLOBAL/ P, QP, K, QK, SVK, SR, SI, U,                 QUA   70
     * V, A, B, C, D, A1, A2, A3, A6, A7, E, F, G,                  QUA   80
     * H, SZR, SZI, LZR, LZI, ETA, ARE, MRE, N, NN                  QUA   90
      DOUBLE PRECISION P(101), QP(101), K(101),                     QUA  100
     * QK(101), SVK(101), SR, SI, U, V, A, B, C, D,                 QUA  110
     * A1, A2, A3, A6, A7, E, F, G, H, SZR, SZI,                    QUA  120
     * LZR, LZI                                                     QUA  130
      REAL ETA, ARE, MRE                                            QUA  140
      INTEGER N, NN                                                 QUA  150
      DOUBLE PRECISION UI, VI, UU, VV, DABS                         QUA  160
      REAL MS, MP, OMP, EE, RELSTP, T, ZM                           QUA  170
      INTEGER NZ, TYPE, I, J                                        QUA  180
      LOGICAL TRIED                                                 QUA  190
      NZ = 0                                                        QUA  200
      TRIED = .FALSE.                                               QUA  210
      U = UU                                                        QUA  220
      V = VV                                                        QUA  230
      J = 0                                                         QUA  240
C MAIN LOOP                                                         QUA  250
   10 CALL QUAD(1.D0, U, V, SZR, SZI, LZR, LZI)                     QUA  260
C RETURN IF ROOTS OF THE QUADRATIC ARE REAL AND NOT                 QUA  270
C CLOSE TO MULTIPLE OR NEARLY EQUAL AND  OF OPPOSITE                QUA  280
C SIGN                                                              QUA  290
      IF (DABS(DABS(SZR)-DABS(LZR)).GT..01D0*                       QUA  300
     * DABS(LZR)) RETURN                                           QUA  310
C EVALUATE POLYNOMIAL BY QUADRATIC SYNTHETIC DIVISION               QUA  320
      CALL QUADSD(NN, U, V, P, QP, A, B)                            QUA  330
      MP = DABS(A-SZR*B) + DABS(SZI*B)                              QUA  340
C COMPUTE A RIGOROUS  BOUND ON THE ROUNDING ERROR IN                QUA  350
C EVALUTING P                                                       QUA  360
      ZM = SQRT(ABS(SNGL(V)))                                       QUA  370
      EE = 2.*ABS(SNGL(QP(1)))                                      QUA  380
      T = -SZR*B                                                    QUA  390
      DO 20 I=2,N                                                   QUA  400
        EE = EE*ZM + ABS(SNGL(QP(I)))                               QUA  410
   20 CONTINUE                                                      QUA  420
      EE = EE*ZM + ABS(SNGL(A)+T)                                   QUA  430
      EE = (5.*MRE+4.*ARE)*EE - (5.*MRE+2.*ARE)*                    QUA  440
     * (ABS(SNGL(A)+T)+ABS(SNGL(B))*ZM) +                         QUA  450
     * 2.*ARE*ABS(T)                                               QUA  460
C ITERATION HAS CONVERGED SUFFICIENTLY IF THE                       QUA  470
C POLYNOMIAL VALUE IS LESS THAN 20 TIMES THIS BOUND                 QUA  480
      IF (MP.GT.20.*EE) GO TO 30                                    QUA  490
      NZ = 2                                                        QUA  500
      RETURN                                                        QUA  510
   30 J = J + 1                                                     QUA  520
C STOP ITERATION AFTER 20 STEPS                                     QUA  530
      IF (J.GT.20) RETURN                                           QUA  540
      IF (J.LT.2) GO TO 50                                          QUA  550
      IF (RELSTP.GT..01 .OR. MP.LT.OMP .OR. TRIED)                  QUA  560
     * GO TO 50                                                     QUA  570
C A CLUSTER APPEARS TO BE STALLING THE CONVERGENCE.                 QUA  580
```

```
C FIVE FIXED SHIFT STEPS ARE TAKEN WITH A U,V CLOSE                    QUA  590
C TO THE CLUSTER                                                       QUA  600
      IF (RELSTP.LT.ETA) RELSTP = ETA                                  QUA  610
      RELSTP = SQRT(RELSTP)                                            QUA  620
      U = U - U*RELSTP                                                 QUA  630
      V = V + V*RELSTP                                                 QUA  640
      CALL QUADSD(NN, U, V, P, QP, A, B)                               QUA  650
      DO 40 I=1,5                                                      QUA  660
        CALL CALCSC(TYPE)                                              QUA  670
        CALL NEXTK(TYPE)                                               QUA  680
   40 CONTINUE                                                         QUA  690
      TRIED = .TRUE.                                                   QUA  700
      J = 0                                                            QUA  710
   50 OMP = MP                                                         QUA  720
C CALCULATE NEXT K POLYNOMIAL AND NEW U AND V                          QUA  730
      CALL CALCSC(TYPE)                                                QUA  740
      CALL NEXTK(TYPE)                                                 QUA  750
      CALL CALCSC(TYPE)                                                QUA  760
      CALL NEWEST(TYPE, UI, VI)                                        QUA  770
C IF VI IS ZERO THE ITERATION IS NOT CONVERGING                        QUA  780
      IF (VI.EQ.0.D0) RETURN                                           QUA  790
      RELSTP = DABS((VI-V)/VI)                                         QUA  800
      U = UI                                                           QUA  810
      V = VI                                                           QUA  820
      GO TO 10                                                         QUA  830
      END                                                              QUA  840


      SUBROUTINE REALIT(SSS, NZ, IFLAG)                                REA   10
C VARIABLE-SHIFT H POLYNOMIAL ITERATION FOR A REAL                     REA   20
C ZERO.                                                                REA   30
C SSS   - STARTING ITERATE                                             REA   40
C NZ    - NUMBER OF ZERO FOUND                                         REA   50
C IFLAG - FLAG TO INDICATE A PAIR OF ZEROS NEAR REAL                   REA   60
C         AXIS.                                                        REA   70
      COMMON /GLOBAL/ P, QP, K, QK, SVK, SR, SI, U,                    REA   80
     * V, A, B, C, D, A1, A2, A3, A6, A7, E, F, G,                    REA   90
     * H, SZR, SZI, LZR, LZI, ETA, ARE, MRE, N, NN                    REA  100
      DOUBLE PRECISION P(101), QP(101), K(101),                        REA  110
     * QK(101), SVK(101), SR, SI, U, V, A, B, C, D,                    REA  120
     * A1, A2, A3, A6, A7, E, F, G, H, SZR, SZI,                       REA  130
     * LZR, LZI                                                        REA  140
      REAL ETA, ARE, MRE                                               REA  150
      INTEGER N, NN                                                    REA  160
      DOUBLE PRECISION PV, KV, T, S, SSS, DABS                         REA  170
      REAL MS, MP, OMP, EE                                             REA  180
      INTEGER NZ, IFLAG, I, J, NM1                                     REA  190
      NM1 = N - 1                                                      REA  200
      NZ = 0                                                           REA  210
      S = SSS                                                          REA  220
      IFLAG = 0                                                        REA  230
      J = 0                                                            REA  240
C MAIN LOOP                                                            REA  250
   10 PV = P(1)                                                        REA  260
C EVALUATE P AT S                                                      REA  270
      QP(1) = PV                                                       REA  280
      DO 20 I=2,NN                                                     REA  290
        PV = PV*S + P(I)                                               REA  300
        QP(I) = PV                                                     REA  310
   20 CONTINUE                                                         REA  320
      MP = DABS(PV)                                                    REA  330
C COMPUTE A RIGOROUS BOUND ON THE ERROR IN EVALUATING                  REA  340
C P                                                                    REA  350
      MS = DABS(S)                                                     REA  360
      EE = (MRE/(ARE+MRE))*ABS(SNGL(QP(1)))                            REA  370
      DO 30 I=2,NN                                                     REA  380
        EE = EE*MS + ABS(SNGL(QP(I)))                                  REA  390
   30 CONTINUE                                                         REA  400
C ITERATION HAS CONVERGED SUFFICIENTLY IF THE                          REA  410
C POLYNOMIAL VALUE IS LESS THAN 20 TIMES THIS BOUND                    REA  420
      IF (MP.GT.20.*((ARE+MRE)*EE-MRE*MP)) GO TO 40                    REA  430
      NZ = 1                                                           REA  440
      SZR = S                                                          REA  450
      SZI = 0.D0                                                       REA  460
      RETURN                                                           REA  470
   40 J = J + 1                                                        REA  480
C STOP ITERATION AFTER 10 STEPS                                        REA  490
      IF (J.GT.10) RETURN                                              REA  500
      IF (J.LT.2) GO TO 50                                             REA  510
      IF (DABS(T).GT..001*DABS(S-T) .OR. MP.LE.OMP)                    REA  520
     * GO TO 50                                                        REA  530
C A CLUSTER OF ZEROS NEAR THE REAL AXIS HAS BEEN                       REA  540
C ENCOUNTERED RETURN WITH IFLAG SET TO INITIATE A                      REA  550
C QUADRATIC ITERATION                                                  REA  560
      IFLAG = 1                                                        REA  570
      SSS = S                                                          REA  580
      RETURN                                                           REA  590
C RETURN IF THE POLYNOMIAL VALUE HAS INCREASED                         REA  600
C SIGNIFICANTLY                                                        REA  610
   50 OMP = MP                                                         REA  620
```

```
C COMPUTE T, THE NEXT POLYNOMIAL, AND THE NEW ITERATE                REA  630
      KV = K(1)                                                      REA  640
      QK(1) = KV                                                     REA  650
      DO 60 I=2,N                                                    REA  660
        KV = KV*S + K(I)                                             REA  670
        QK(I) = KV                                                   REA  680
   60 CONTINUE                                                       REA  690
      IF (DABS(KV).LE.DABS(K(N))*10.*ETA) GO TO 80                   REA  700
C USE THE SCALED FORM OF THE RECURRENCE IF THE VALUE                 REA  710
C OF K AT S IS NONZERO                                               REA  720
      T = -PV/KV                                                     REA  730
      K(1) = QP(1)                                                   REA  740
      DO 70 I=2,N                                                    REA  750
        K(I) = T*QK(I-1) + QP(I)                                     REA  760
   70 CONTINUE                                                       REA  770
      GO TO 100                                                      REA  780
C USE UNSCALED FORM                                                  REA  790
   80 K(1) = 0.0D0                                                   REA  800
      DO 90 I=2,N                                                    REA  810
        K(I) = QK(I-1)                                               REA  820
   90 CONTINUE                                                       REA  830
  100 KV = K(1)                                                      REA  840
      DO 110 I=2,N                                                   REA  850
        KV = KV*S + K(I)                                             REA  860
  110 CONTINUE                                                       REA  870
      T = 0.D0                                                       REA  880
      IF (DABS(KV).GT.DABS(K(N))*10.*ETA) T = -PV/KV                 REA  890
      S = S + T                                                      REA  900
      GO TO 10                                                       REA  910
      END                                                            REA  920


      SUBROUTINE CALCSC(TYPE)                                        CAL   10
C THIS ROUTINE CALCULATES SCALAR QUANTITIES USED TO                  CAL   20
C COMPUTE THE NEXT K POLYNOMIAL AND NEW ESTIMATES OF                 CAL   30
C THE QUADRATIC COEFFICIENTS.                                        CAL   40
C TYPE - INTEGER VARIABLE SET HERE INDICATING HOW THE                CAL   50
C CALCULATIONS ARE NORMALIZED TO AVOID OVERFLOW                      CAL   60
      COMMON /GLOBAL/ P, QP, K, QK, SVK, SR, SI, U,                  CAL   70
     * V, A, B, C, D, A1, A2, A3, A6, A7, E, F, G,                  CAL   80
     * H, SZR, SZI, LZR, LZI, ETA, ARE, MRE, N, NN                  CAL   90
      DOUBLE PRECISION P(101), QP(101), K(101),                      CAL  100
     * QK(101), SVK(101), SR, SI, U, V, A, B, C, D,                 CAL  110
     * A1, A2, A3, A6, A7, E, F, G, H, SZR, SZI,                    CAL  120
     * LZR, LZI                                                     CAL  130
      REAL ETA, ARE, MRE                                             CAL  140
      INTEGER N, NN                                                  CAL  150
      DOUBLE PRECISION DABS                                          CAL  160
      INTEGER TYPE                                                   CAL  170
C SYNTHETIC DIVISION OF K BY THE QUADRATIC 1,U,V                     CAL  180
      CALL QUADSD(N, U, V, K, QK, C, D)                              CAL  190
      IF (DABS(C).GT.DABS(K(N))*100.*ETA) GO TO 10                   CAL  200
      IF (DABS(D).GT.DABS(K(N-1))*100.*ETA) GO TO 10                 CAL  210
      TYPE = 3                                                       CAL  220
C TYPE=3 INDICATES THE QUADRATIC IS ALMOST A FACTOR                  CAL  230
C OF K                                                               CAL  240
      RETURN                                                         CAL  250
   10 IF (DABS(D).LT.DABS(C)) GO TO 20                               CAL  260
      TYPE = 2                                                       CAL  270
C TYPE=2 INDICATES THAT ALL FORMULAS ARE DIVIDED BY D                CAL  280
      E = A/D                                                        CAL  290
      F = C/D                                                        CAL  300
      G = U*B                                                        CAL  310
      H = V*B                                                        CAL  320
      A3 = (A+G)*E + H*(B/D)                                         CAL  330
      A1 = B*F - A                                                   CAL  340
      A7 = (F+U)*A + H                                               CAL  350
      RETURN                                                         CAL  360
   20 TYPE = 1                                                       CAL  370
C TYPE=1 INDICATES THAT ALL FORMULAS ARE DIVIDED BY C                CAL  380
      E = A/C                                                        CAL  390
      F = D/C                                                        CAL  400
      G = U*E                                                        CAL  410
      H = V*B                                                        CAL  420
      A3 = A*E + (H/C+G)*B                                           CAL  430
      A1 = B - A*(D/C)                                               CAL  440
      A7 = A + G*D + H*F                                             CAL  450
      RETURN                                                         CAL  460
      END                                                            CAL  470


      SUBROUTINE NEXTK(TYPE)                                         NEX   10
C COMPUTES THE NEXT K POLYNOMIALS USING SCALARS                      NEX   20
C COMPUTED IN CALCSC                                                 NEX   30
      COMMON /GLOBAL/ P, QP, K, QK, SVK, SR, SI, U,                  NEX   40
     * V, A, B, C, D, A1, A2, A3, A6, A7, E, F, G,                  NEX   50
     * H, SZR, SZI, LZR, LZI, ETA, ARE, MRE, N, NN                  NEX   60
      DOUBLE PRECISION P(101), QP(101), K(101),                      NEX   70
```

```
     * QK(101), SVK(101), SR, SI, U, V, A, B, C, D,                    NEX   80
     * A1, A2, A3, A6, A7, E, F, G, H, SZR, SZI,                       NEX   90
     * LZR, LZI                                                        NEX  100
      REAL ETA, ARE, MRE                                              NEX  110
      INTEGER N, NN                                                   NEX  120
      DOUBLE PRECISION TEMP, DABS                                     NEX  130
      INTEGER TYPE                                                    NEX  140
      IF (TYPE.EQ.3) GO TO 40                                         NEX  150
      TEMP = A                                                        NEX  160
      IF (TYPE.EQ.1) TEMP = B                                         NEX  170
      IF (DABS(A1).GT.DABS(TEMP)*ETA*10.) GO TO 20                    NEX  180
C IF A1 IS NEARLY ZERO THEN USE A SPECIAL FORM OF THE                 NEX  190
C RECURRENCE                                                          NEX  200
      K(1) = 0.D0                                                     NEX  210
      K(2) = -A7*QP(1)                                                NEX  220
      DO 10 I=3,N                                                     NEX  230
        K(I) = A3*QK(I-2) - A7*QP(I-1)                                NEX  240
   10 CONTINUE                                                        NEX  250
      RETURN                                                          NEX  260
C USE SCALED FORM OF THE RECURRENCE                                   NEX  270
   20 A7 = A7/A1                                                      NEX  280
      A3 = A3/A1                                                      NEX  290
      K(1) = QP(1)                                                    NEX  300
      K(2) = QP(2) - A7*QP(1)                                         NEX  310
      DO 30 I=3,N                                                     NEX  320
        K(I) = A3*QK(I-2) - A7*QP(I-1) + QP(I)                        NEX  330
   30 CONTINUE                                                        NEX  340
      RETURN                                                          NEX  350
C USE UNSCALED FORM OF THE RECURRENCE IF TYPE IS 3                    NEX  360
   40 K(1) = 0.D0                                                     NEX  370
      K(2) = 0.D0                                                     NEX  380
      DO 50 I=3,N                                                     NEX  390
        K(I) = QK(I-2)                                                NEX  400
   50 CONTINUE                                                        NEX  410
      RETURN                                                          NEX  420
      END                                                             NEX  430


      SUBROUTINE NEWEST(TYPE, UU, VV)                                 NEW   10
C COMPUTE NEW ESTIMATES OF THE QUADRATIC COEFFICIENTS                 NEW   20
C USING THE SCALARS COMPUTED IN CALCSC.                               NEW   30
      COMMON /GLOBAL/ P, QP, K, QK, SVK, SR, SI, U,                   NEW   40
     * V, A, B, C, D, A1, A2, A3, A6, A7, E, F, G,                    NEW   50
     * H, SZR, SZI, LZR, LZI, ETA, ARE, MRE, N, NN                    NEW   60
      DOUBLE PRECISION P(101), QP(101), K(101),                       NEW   70
     * QK(101), SVK(101), SR, SI, U, V, A, B, C, D,                   NEW   80
     * A1, A2, A3, A6, A7, E, F, G, H, SZR, SZI,                      NEW   90
     * LZR, LZI                                                       NEW  100
      REAL ETA, ARE, MRE                                              NEW  110
      INTEGER N, NN                                                   NEW  120
      DOUBLE PRECISION A4, A5, B1, B2, C1, C2, C3,                    NEW  130
     * C4, TEMP, UU, VV                                               NEW  140
      INTEGER TYPE                                                    NEW  150
C USE FORMULAS APPROPRIATE TO SETTING OF TYPE.                        NEW  160
      IF (TYPE.EQ.3) GO TO 30                                         NEW  170
      IF (TYPE.EQ.2) GO TO 10                                         NEW  180
      A4 = A + U*B + H*F                                              NEW  190
      A5 = C + (U+V*F)*D                                              NEW  200
      GO TO 20                                                        NEW  210
   10 A4 = (A+G)*F + H                                                NEW  220
      A5 = (F+U)*C + V*D                                              NEW  230
C EVALUATE NEW QUADRATIC COEFFICIENTS.                                NEW  240
   20 B1 = -K(N)/P(NN)                                                NEW  250
      B2 = -(K(N-1)+B1*P(N))/P(NN)                                    NEW  260
      C1 = V*B2*A1                                                    NEW  270
      C2 = B1*A7                                                      NEW  280
      C3 = B1*B1*A3                                                   NEW  290
      C4 = C1 - C2 - C3                                               NEW  300
      TEMP = A5 + B1*A4 - C4                                          NEW  310
      IF (TEMP.EQ.0.D0) GO TO 30                                      NEW  320
      UU = U - (U*(C3+C2)+V*(B1*A1+B2*A7))/TEMP                       NEW  330
      VV = V*(1.+C4/TEMP)                                             NEW  340
      RETURN                                                          NEW  350
C IF TYPE=3 THE QUADRATIC IS ZEROED                                   NEW  360
   30 UU = 0.D0                                                       NEW  370
      VV = 0.D0                                                       NEW  380
      RETURN                                                          NEW  390
      END                                                             NEW  400


      SUBROUTINE QUADSD(NN, U, V, P, Q, A, B)                         QUA   10
C DIVIDES P BY THE QUADRATIC  1,U,V  PLACING THE                      QUA   20
C QUOTIENT IN Q AND THE REMAINDER IN A,B                              QUA   30
      DOUBLE PRECISION P(NN), Q(NN), U, V, A, B, C                    QUA   40
      INTEGER I                                                       QUA   50
      B = P(1)                                                        QUA   60
      Q(1) = B                                                        QUA   70
      A = P(2) - U*B                                                  QUA   80
      Q(2) = A                                                        QUA   90
      DO 10 I=3,NN                                                    QUA  100
        C = P(I) - U*A - V*B                                          QUA  110
```

```
        Q(I) = C                                                    QUA  120
        B = A                                                       QUA  130
        A = C                                                       QUA  140
   10 CONTINUE                                                      QUA  150
      RETURN                                                        QUA  160
      END                                                           QUA  170
      SUBROUTINE QUAD(A, B1, C, SR, SI, LR, LI)                     QUA   10
C CALCULATE THE ZEROS OF THE QUADRATIC A*Z**2+B1*Z+C.               QUA   20
C THE QUADRATIC FORMULA, MODIFIED TO AVOID                          QUA   30
C OVERFLOW, IS USED TO FIND THE LARGER ZERO IF THE                  QUA   40
C ZEROS ARE REAL AND BOTH ZEROS ARE COMPLEX.                        QUA   50
C THE SMALLER REAL ZERO IS FOUND DIRECTLY FROM THE                  QUA   60
C PRODUCT OF THE ZEROS C/A.                                         QUA   70
      DOUBLE PRECISION A, B1, C, SR, SI, LR, LI, B,                 QUA   80
     * D, E, DABS, DSQRT                                           QUA   90
      IF (A.NE.0.D0) GO TO 20                                       QUA  100
      SR = 0.D0                                                     QUA  110
      IF (B1.NE.0.D0) SR = -C/B1                                    QUA  120
      LR = 0.D0                                                     QUA  130
   10 SI = 0.D0                                                     QUA  140
      LI = 0.D0                                                     QUA  150
      RETURN                                                        QUA  160
   20 IF (C.NE.0.D0) GO TO 30                                       QUA  170
      SR = 0.D0                                                     QUA  180
      LR = -B1/A                                                    QUA  190
      GO TO 10                                                      QUA  200
C COMPUTE DISCRIMINANT AVOIDING OVERFLOW                            QUA  210
   30 B = B1/2.D0                                                   QUA  220
      IF (DABS(B).LT.DABS(C)) GO TO 40                              QUA  230
      E = 1.D0 - (A/B)*(C/B)                                        QUA  240
      D = DSQRT(DABS(E))*DABS(B)                                    QUA  250
      GO TO 50                                                      QUA  260
   40 E = A                                                         QUA  270
      IF (C.LT.0.D0) E = -A                                         QUA  280
      E = B*(B/DABS(C)) - E                                         QUA  290
      D = DSQRT(DABS(E))*DSQRT(DABS(C))                             QUA  300
   50 IF (E.LT.0.D0) GO TO 60                                       QUA  310
C REAL ZEROS                                                        QUA  320
      IF (B.GE.0.D0) D = -D                                         QUA  330
      LR = (-B+D)/A                                                 QUA  340
      SR = 0.D0                                                     QUA  350
      IF (LR.NE.0.D0) SR = (C/LR)/A                                 QUA  360
      GO TO 10                                                      QUA  370
C COMPLEX CONJUGATE ZEROS                                           QUA  380
   60 SR = -B/A                                                     QUA  390
      LR = SR                                                       QUA  400
      SI = DABS(D/A)                                                QUA  410
      LI = -SI                                                      QUA  420
      RETURN                                                        QUA  430
      END                                                           QUA  440
```

ALGORITHM 494
PDEONE, Solutions of Systems of Partial Differential Equations [D3]

RICHARD F. SINCOVEC

Kansas State University

and

NIEL K. MADSEN

Lawrence Livermore Laboratory

Key Words and Phrases: partial differential equations, method of lines, ordinary differential equations
CR Categories: 5.17
Language: Fortran

DESCRIPTION

The description of this algorithm, test results, and references are contained in the authors' paper, "Software for Nonlinear Partial Differential Equations," *ACM Trans. Math. Software 1*, 3 (Sept. 1975), 232–260.

ALGORITHM

```
      SUBROUTINE PDEONE(T, U, UDOT, NPDE, NPTS)                          PDE   10
      DIMENSION U(NPDE,NPTS), UDOT(NPDE,NPTS)                            PDE   20
C PDEONE IS AN INTERFACE SUBROUTINE WHICH USES CENTERED DIF-             PDE   30
C FERENCE APPROXIMATIONS TO CONVERT ONE-DIMENSIONAL SYSTEMS              PDE   40
C OF PARTIAL DIFFERENTIAL EQUATIONS INTO A SYSTEM OF ORDINARY            PDE   50
C DIFFERENTIAL EQUATIONS, UDOT = F(T,X,U).  THIS ROUTINE IS              PDE   60
C INTENDED TO BE USED WITH A ROBUST ODE INTEGRATOR.                      PDE   70
C INPUT..                                                                PDE   80
C  NPDE = NUMBER OF PARTIAL DIFFERENTIAL EQUATIONS.                      PDE   90
C  NPTS = NUMBER OF SPATIAL GRID POINTS.                                 PDE  100
C     T = CURRENT VALUE OF TIME.                                         PDE  110
C     U = AN NPDE BY NPTS ARRAY CONTAINING THE COMPUTED                  PDE  120
C         SOLUTION AT TIME T.                                            PDE  130
C OUTPUT..                                                               PDE  140
C  UDOT = AN NPDE BY NPTS ARRAY CONTAINING THE RIGHT HAND                PDE  150
C         SIDE OF THE RESULTING SYSTEM OF ODE*S, F(T,X,U),               PDE  160
C         OBTAINED BY DISCRETIZING THE GIVEN PDE*S.                      PDE  170
C THE USER MUST INSERT A DIMENSION STATEMENT OF THE FOLLOW-              PDE  180
C ING FORM..                                                             PDE  190
C     DIMENSION DVAL(**,**,2),UX(**),UAVG(**),ALPHA(**),                 PDE  200
C    *          BETA(**),GAMMA(**)                                       PDE  210
C THE SYMBOLS ** DENOTE THE ACTUAL NUMERICAL VALUE OF NPDE               PDE  220
C FOR THE PROBLEM BEING SOLVED.                                          PDE  230
C COMMON BLOCK MESH CONTAINS THE USER SPECIFIED SPATIAL                  PDE  240
C GRID POINTS.                                                           PDE  250
C COMMON BLOCK COORD CONTAINS 0,1, OR 2 DEPENDING ON WHETHER             PDE  260
C THE PROBLEM IS IN CARTESIAN, CYLINDRICAL, OR SPHERICAL                 PDE  270
```

Received 1 July 1974 and 4 February 1974.

Authors' addresses: R.F. Sincovec, Kansas State University, Manhattan, KS 66506; N.K. Madsen, Lawrence Livermore Laboratory, Livermore, CA 94550.

```
C COORDINATES, RESPECTIVELY.                                                PDE  280
      COMMON /MESH/ X(1)                                                    PDE  290
      COMMON /COORD/ ICORD                                                  PDE  300
      ICORD1 = ICORD + 1                                                    PDE  310
C UPDATE THE LEFT BOUNDARY VALUES                                           PDE  320
      CALL BNDRY(T, X(1), U, ALPHA, BETA, GAMMA, NPDE)                      PDE  330
      ITEST = 0                                                             PDE  340
      DXI = 1./(X(2)-X(1))                                                  PDE  350
      DO 10 K=1,NPDE                                                        PDE  360
        IF (BETA(K).NE.0.0) GO TO 10                                        PDE  370
        U(K,1) = GAMMA(K)/ALPHA(K)                                          PDE  380
        ITEST = ITEST + 1                                                   PDE  390
   10 CONTINUE                                                              PDE  400
      IF (ITEST.EQ.NPDE) GO TO 50                                           PDE  410
      IF (ITEST.EQ.0) GO TO 20                                              PDE  420
      CALL BNDRY(T, X(1), U, ALPHA, BETA, GAMMA, NPDE)                      PDE  430
C EVALUATE D COEFFICIENTS AT THE LEFT BOUNDARY                              PDE  440
   20 CALL D(T, X(1), U, DVAL, NPDE)                                        PDE  450
C FORM APPROXIMATIONS TO DU/DX AT THE LEFT BOUNDARY                         PDE  460
      DO 40 K=1,NPDE                                                        PDE  470
        IF (BETA(K).NE.0.0) GO TO 30                                        PDE  480
        UX(K) = DXI*(U(K,2)-U(K,1))                                         PDE  490
        GO TO 40                                                            PDE  500
   30   UX(K) = (GAMMA(K)-ALPHA(K)*U(K,1))/BETA(K)                          PDE  510
   40 CONTINUE                                                              PDE  520
C EVALUATE U-AVERAGE IN THE FIRST INTERVAL                                  PDE  530
   50 DO 60 K=1,NPDE                                                        PDE  540
        UAVG(K) = .5*(U(K,2)+U(K,1))                                        PDE  550
   60 CONTINUE                                                              PDE  560
C EVALUATE D COEFFICIENTS IN THE FIRST INTERVAL                             PDE  570
      XAVGR = .5*(X(2)+X(1))                                                PDE  580
      CALL D(T, XAVGR, UAVG, DVAL(1,1,2), NPDE)                             PDE  590
      DXIL = 1.                                                             PDE  600
      DXIR = DXI                                                            PDE  610
      IF (ICORD.EQ.0) GO TO 70                                              PDE  620
      DXIL = X(1)**ICORD                                                    PDE  630
      DXIR = DXIR*XAVGR**ICORD                                              PDE  640
C EVALUATE DUXX AT THE LEFT BOUNDARY                                        PDE  650
   70 IF (ITEST.EQ.NPDE) GO TO 100                                          PDE  660
      DXIC = FLOAT(ICORD1)/(XAVGR**ICORD1-X(1)**ICORD1)                     PDE  670
      DO 90 L=1,NPDE                                                        PDE  680
        DO 80 K=1,NPDE                                                      PDE  690
          DVAL(K,L,1) = DXIC*(DVAL(K,L,2)*(U(L,2)-U(L,1))*                  PDE  700
     *      DXIR-DVAL(K,L,1)*UX(L)*DXIL)                                    PDE  710
   80   CONTINUE                                                            PDE  720
   90 CONTINUE                                                              PDE  730
C EVALUATE RIGHTHAND SIDE OF PDE*S AT THE LEFT BOUNDARY                     PDE  740
      CALL F(T, X(1), U, UX, DVAL, UDOT, NPDE)                              PDE  750
C SET UDOT = 0 FOR KNOWN LEFT BOUNDARY VALUES                               PDE  760
  100 DO 110 K=1,NPDE                                                       PDE  770
        IF (BETA(K).EQ.0.0) UDOT(K,1) = 0.0                                 PDE  780
  110 CONTINUE                                                              PDE  790
C UPDATE THE RIGHT BOUNDARY VALUES                                          PDE  800
      CALL BNDRY(T, X(NPTS), U(1,NPTS), ALPHA, BETA, GAMMA, NPDE)           PDE  810
      ITEST = 0                                                             PDE  820
      DO 120 K=1,NPDE                                                       PDE  830
        IF (BETA(K).NE.0.0) GO TO 120                                       PDE  840
        U(K,NPTS) = GAMMA(K)/ALPHA(K)                                       PDE  850
        ITEST = ITEST + 1                                                   PDE  860
  120 CONTINUE                                                              PDE  870
      IBCK = 1                                                              PDE  880
      IFWD = 2                                                              PDE  890
      ILIM = NPTS - 1                                                       PDE  900
C MAIN LOOP TO FORM ORDINARY DIFFERENTIAL EQUATIONS AT THE                  PDE  910
C GRID POINTS                                                               PDE  920
      DO 160 I=2,ILIM                                                       PDE  930
        K = IBCK                                                            PDE  940
        IBCK = IFWD                                                         PDE  950
        IFWD = K                                                            PDE  960
        XAVGL = XAVGR                                                       PDE  970
        XAVGR = .5*(X(I+1)+X(I))                                            PDE  980
        DXI = 1./(X(I+1)-X(I-1))                                            PDE  990
        DXIL = DXIR                                                         PDE 1000
        DXIR = 1./(X(I+1)-X(I))                                             PDE 1010
        IF (ICORD.NE.0) DXIR = DXIR*XAVGR**ICORD                            PDE 1020
        DXIC = FLOAT(ICORD1)/(XAVGR**ICORD1-XAVGL**ICORD1)                  PDE 1030
C EVALUATE DU/DX AND U-AVERAGE AT THE I-TH GRID POINT AND                   PDE 1040
C INTERVAL RESPECTIVELY                                                     PDE 1050
        DO 130 K=1,NPDE                                                     PDE 1060
          UX(K) = DXI*(U(K,I+1)-U(K,I-1))                                   PDE 1070
          UAVG(K) = .5*(U(K,I+1)+U(K,I))                                    PDE 1080
  130   CONTINUE                                                            PDE 1090
C EVALUATE D COEFFICIENTS IN THE I-TH INTERVAL                              PDE 1100
        CALL D(T, XAVGR, UAVG, DVAL(1,1,IFWD), NPDE)                        PDE 1110
C EVALUATE DUXX AT THE I-TH GRID POINT                                      PDE 1120
        DO 150 L=1,NPDE                                                     PDE 1130
          DO 140 K=1,NPDE                                                   PDE 1140
            DVAL(K,L,IBCK) = (DVAL(K,L,IFWD)*(U(L,I+1)-U(L,I))*             PDE 1150
     *        DXIR-DVAL(K,L,IBCK)*(U(L,I)-U(L,I-1))*DXIL)*DXIC              PDE 1160
  140     CONTINUE                                                          PDE 1170
  150   CONTINUE                                                            PDE 1180
```

```
C EVALUATE RIGHTHAND SIDE OF PDE*S AT THE I-TH GRID POINT                PDE 1190
       CALL F(T, X(I), U(1,I), UX, DVAL(1,1,IBCK), UDOT(1,I), NPDE)      PDE 1200
  160 CONTINUE                                                           PDE 1210
C FINISH UPDATING THE RIGHT BOUNDARY IF NECESSARY                        PDE 1220
      IF (ITEST.EQ.NPDE) GO TO 220                                       PDE 1230
      IF (ITEST.EQ.0) GO TO 170                                          PDE 1240
      CALL BNDRY(T, X(NPTS), U(1,NPTS), ALPHA, BETA, GAMMA, NPDE)        PDE 1250
C EVALUATE D COEFFICIENTS AT THE RIGHT BOUNDARY                          PDE 1260
  170 CALL D(T, X(NPTS), U(1,NPTS), DVAL(1,1,IBCK), NPDE)                PDE 1270
C FORM APPROXIMATIONS TO DU/DX AT THE RIGHT BOUNDARY                     PDE 1280
      DXI = 1./(X(NPTS)-X(ILIM))                                         PDE 1290
      DO 190 K=1,NPDE                                                    PDE 1300
        IF (BETA(K).NE.0.0) GO TO 180                                    PDE 1310
        UX(K) = DXI*(U(K,NPTS)-U(K,ILIM))                                PDE 1320
        GO TO 190                                                        PDE 1330
  180   UX(K) = (GAMMA(K)-ALPHA(K)*U(K,NPTS))/BETA(K)                   PDE 1340
  190 CONTINUE                                                           PDE 1350
      DXIL = DXIR                                                        PDE 1360
      DXIR = 1.                                                          PDE 1370
      IF (ICORD.NE.0) DXIR = X(NPTS)**ICORD                              PDE 1380
      DXIC = FLOAT(ICORD1)/(X(NPTS)**ICORD1-XAVGR**ICORD1)               PDE 1390
C EVALUATE DUXX AT THE RIGHT BOUNDARY                                    PDE 1400
      DO 210 L=1,NPDE                                                    PDE 1410
        DO 200 K=1,NPDE                                                  PDE 1420
          DVAL(K,L,IBCK) = DXIC*(DVAL(K,L,IBCK)*UX(L)*DXIR-DVAL(K,L,     PDE 1430
     *      IFWD)*(U(L,NPTS)-U(L,ILIM))*DXIL)                            PDE 1440
  200   CONTINUE                                                         PDE 1450
  210 CONTINUE                                                           PDE 1460
C EVALUATE RIGHTHAND SIDE OF PDE*S AT THE RIGHT BOUNDARY                 PDE 1470
      CALL F(T, X(NPTS), U(1,NPTS), UX, DVAL(1,1,IBCK),                  PDE 1480
     * UDOT(1,NPTS), NPDE)                                              PDE 1490
C SET UDOT = 0 FOR KNOWN RIGHT BOUNDARY VALUES                           PDE 1500
  220 DO 230 K=1,NPDE                                                    PDE 1510
        IF (BETA(K).EQ.0.0) UDOT(K,NPTS) = 0.0                           PDE 1520
  230 CONTINUE                                                           PDE 1530
      RETURN                                                             PDE 1540
      END                                                                PDE 1550
```

ALGORITHM 495
Solution of an Overdetermined System of Linear Equations in the Chebyshev Norm [F4]

I. BARRODALE

University of Victoria, Canada

and

C. PHILLIPS

University of Liverpool, England

Key Words and Phrases: Chebyshev solution, overdetermined system, linear programming, simplex method
CR Categories: 3.15
Language: Fortran

DESCRIPTION

The algorithm calculates a Chebyshev (or l_∞) solution to an $m \times n$ overdetermined system of linear equations $Ax = b$, i.e. given equations

$$\sum_{j=1}^{n} a_{i,j} x_j = b_i , \qquad i = 1, 2, \ldots, \mathrm{m},\ m \geq n,$$

the subroutine determines a vector $x^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ which minimizes the maximum absolute value of the residuals

$$e(x) = || b - Ax ||_\infty = \max_{1 \leq i \leq m} \left| b_i - \sum_{j=1}^{n} a_{i,j} x_j \right| . \tag{1}$$

The algorithm can be used to solve the linear Chebyshev data fitting problem. Suppose that data consisting of m points (z_i, y_i) are to be approximated by a linear approximating function $\alpha_1\phi_1(z) + \alpha_2\phi_2(z) + \cdots + \alpha_n\phi_n(z)$ in the Chebyshev norm. This is equivalent to finding a Chebyshev solution to the system of linear equations

$$\sum_{j=1}^{n} \phi_j(z_i)\alpha_j = y_i , \qquad i = 1, 2, \ldots, m.$$

If the y_i values contain only small inherent errors, a Chebyshev approximation may be preferred to a least-squares or least-first-power approximation.

Received 3 June 1974 and 28 January 1975.

Authors' addresses: I. Barrodale, Department of Mathematics, University of Victoria, Victoria, B.C., Canada; C. Phillips, Department of Computational and Statistical Science, University of Liverpool, Liverpool, England.

The algorithm is a modification of the simplex method of linear programming applied to the *dual* formulation of the Chebyshev problem. It does not require that the $m \times n$ matrix of coefficients $A = \{a_{i,j}\}$ satisfy the Haar condition, nor does it require that it be of full rank. Computational experience with this and other algorithms indicates that it is efficient in both time and storage requirements. An approximate solution is determined during the first $k + 1$ iterations, where k denotes the rank of the matrix A, and in the subsequent iterations this solution is refined until either (i) a Chebyshev solution x^* is obtained which minimizes $e(x)$, or (ii) an approximate solution $\hat{x}$ is found for which

$$(e(\hat{x}) - e(x^*))/e(x^*) < RELERR \tag{2}$$

where *RELERR* is a real variable specified by the user (see below).

The parameters M and N represent the number of equations and number of unknowns, respectively. *MDIM* and *NDIM* should be set to at least $M + 1$ and $N + 3$, respectively. The simplex iterations are carried out in the two-dimensional real array A of size (*NDIM*, *MDIM*). Initially, the *transpose* of the matrix A must be stored in the first N rows and M columns of A; this arrangement allows the code to correspond closely to the formal description of the algorithm [1]. The right-hand-side vector b must be stored in the array B. These initial coefficients assigned to A and B are subsequently destroyed. *TOL* is a real variable which should be set to a small positive value; e.g. $TOL = 10^{-d+1}$ where d represents the number of decimal digits of accuracy available. (On an IBM 370 we usually set *TOL* equal to 10^{-6} when using single precision arithmetic and equal to 10^{-15} when using double precision arithmetic.) Essentially, *the subroutine cannot distinguish between zero and any quantity whose magnitude does not exceed TOL.* In particular, it will not pivot on any number whose magnitude does not exceed *TOL.* (Consequently, the subroutine may terminate prematurely if the system of equations is poorly scaled.) The parameter *RELERR* should be set to 0.0 if a Chebyshev solution x^* is required. If *RELERR* is set to a positive value, the subroutine determines an approximate solution $\hat{x}$ satisfying inequality (2); the usual effect of this option (say, with *RELERR* set to 0.1) is a saving in the number of simplex iterations required.

On exit, if *RELERR* was set to 0.0, the array X contains a Chebyshev solution x^*. However, if the initial value of *RELERR* was positive, then X contains an approximate solution $\hat{x}$. The subroutine assigns a new value to *RELERR*, smaller than its initial value, and $\hat{x}$ satisfies (2) for this new value of *RELERR*. The rank of the $m \times n$ matrix A is determined during the first k iterations and assigned to the integer *RANK*. (The rank estimate *RANK* is dependent upon *TOL*; decreasing the value of *TOL* may increase the value of *RANK*.) The residuals

$$b_i - \sum_{j=1}^{n} a_{i,j} x_j, \qquad i = 1, 2, \ldots, m \tag{3}$$

are assigned in order to the array B. *RESMAX* supplies the absolute value of the largest residual; this is the minimum value of (1) if *RELERR* was set to 0.0. If *RELERR* was set to a positive value, then *RESMAX* gives the value of (1) corresponding to the approximate solution $\hat{x}$. (The subroutine, which overwrites the original data A and b, supplies the residuals automatically. However, these automatic residuals may contain few significant figures, especially when *RESMAX* is within one or two orders of magnitude of *TOL.* Indeed, if $RESMAX \leq TOL$ the automatic residuals may all be set to zero by the subroutine. *It is therefore often advisable to obtain the residuals directly from* (3) *as an additional calculation.*) *ITER* records the number of iterations performed by the simplex method. Finally, *OCODE* is an exit code with the value 1 if a solution is calculated successfully, and 2 if the calculations are terminated prematurely. This latter condition occurs only when a pivot is encountered whose magnitude does not exceed *TOL*, and in this event all output information pertains to the last completed simplex iteration. Since a

Chebyshev solution is not necessarily unique, the subroutine attempts to determine if other optimal solutions exist. An exit code of 1 indicates that the solution is unique, while an exit code of 0 indicates that the solution is almost certainly not unique. (This uncertainty can only be resolved by a close examination of the final simplex tableau contained in A, and we do not consider such an examination to be warranted in practice). A Chebyshev solution may be nonunique simply because the matrix of coefficients A is not of full rank.

Example. Approximate $y(z) = e^z$ by $\alpha_1 + \alpha_2 z + \alpha_3 z^2 + \alpha_4 z^3$ in the Chebyshev norm on the 51 points defined by $z = 0(0.02)1$. Equivalently, calculate the Chebyshev solution to the system of equations $\sum_{j=1}^{4} z_i^{j-1} x_j = e^{z_i}$, $i = 1, 2, \ldots, 51$, where $z_i = 0.02(i - 1)$. Setting *TOL* equal to 10^{-6} and setting *RELERR* to 0.0, the subroutine gives the following solution on an IBM 370/145 using single precision arithmetic (approximately 7 decimal digits): $x_1^* = 0.999456$, $x_2^* = 1.016599$, $x_3^* = 0.421690$, $x_4^* = 0.279989$. The number of iterations required is 8, and the value of *RESMAX* is 0.000544.

REFERENCES

1. Barrodale, I., and Phillips, C. An improved algorithm for discete Chebyshev linear approximations. Proc. 4th Manitoba Conf. on Numerical Mathematics, U. of Manitoba, Winnipeg, Canada, 1974, pp. 177–190.

ALGORITHM

```
      SUBROUTINE CHEB(M, N, MDIM, NDIM, A, B, TOL, RELERR, X, RANK,      CHE   10
     * RESMAX, ITER, OCODE)                                              CHE   20
C THIS SUBROUTINE USES A MODIFICATION OF THE SIMPLEX METHOD              CHE   30
C OF LINEAR PROGRAMMING TO CALCULATE A CHEBYSHEV SOLUTION TO             CHE   40
C AN OVER-DETERMINED SYSTEM OF LINEAR EQUATIONS.                         CHE   50
C DESCRIPTION OF PARAMETERS.                                             CHE   60
C M       NUMBER OF EQUATIONS.                                           CHE   70
C N       NUMBER OF UNKNOWNS (N MUST NOT EXCEED M).                      CHE   80
C MDIM    THE NUMBER OF COLUMNS OF A, AT LEAST M+1.                      CHE   90
C NDIM    THE NUMBER OF ROWS OF A, AT LEAST N+3.                         CHE  100
C A       TWO DIMENSIONAL REAL ARRAY OF SIZE (NDIM,MDIM).                CHE  110
C         ON ENTRY,THE TRANSPOSE OF THE MATRIX OF                        CHE  120
C         COEFFICIENTS OF THE OVER-DETERMINED SYSTEM MUST                CHE  130
C         BE STORED IN THE FIRST M COLUMNS AND N ROWS OF A.              CHE  140
C         THESE VALUES ARE DESTROYED BY THE SUBROUTINE.                  CHE  150
C B       ONE DIMENSIONAL REAL ARRAY OF SIZE MDIM. ON ENTRY,             CHE  160
C         B MUST CONTAIN THE RIGHT-HAND SIDES OF THE                     CHE  170
C         EQUATIONS IN ITS FIRST M LOCATIONS. ON EXIT, B                 CHE  180
C         CONTAINS THE RESIDUALS FOR THE EQUATIONS IN ITS                CHE  190
C         FIRST M LOCATIONS (SEE DESCRIPTION).                           CHE  200
C TOL     A SMALL POSITIVE TOLERANCE. EMPIRICAL EVIDENCE                 CHE  210
C         SUGGESTS TOL=10**(-D+1) WHERE D REPRESENTS THE                 CHE  220
C         NUMBER OF DECIMAL DIGITS OF ACCURACY AVAILABLE                 CHE  230
C         (SEE DESCRIPTION).                                             CHE  240
C RELERR A REAL VARIABLE WHICH ON ENTRY MUST HAVE THE VALUE              CHE  250
C         0.0 IF A CHEBYSHEV SOLUTION IS REQUIRED. IF RELERR             CHE  260
C         IS POSITIVE, THE SUBROUTINE CALCULATES AN                      CHE  270
C         APPROXIMATE SOLUTION WITH RELERR AS AN UPPER BOUND             CHE  280
C         ON THE RELATIVE ERROR OF ITS LARGEST RESIDUAL (SEE             CHE  290
C         DESCRIPTION-INEQUALITY (2)). ON EXIT, THE VALUE OF             CHE  300
C         RELERR GIVES A SMALLER UPPER BOUND FOR THIS                    CHE  310
C         RELATIVE ERROR.                                                CHE  320
C X       ONE DIMENSIONAL REAL ARRAY OF SIZE NDIM. ON EXIT,              CHE  330
C         THIS ARRAY CONTAINS A SOLUTION TO THE PROBLEM IN               CHE  340
C         ITS FIRST N LOCATIONS.                                         CHE  350
C RANK    AN INTEGER WHICH GIVES ON EXIT THE RANK OF THE                 CHE  360
C         MATRIX OF COEFFICIENTS.                                        CHE  370
C RESMAX THE LARGEST RESIDUAL IN MAGNITUDE.                              CHE  380
C ITER    THE NUMBER OF SIMPLEX ITERATIONS PERFORMED.                    CHE  390
C OCODE   AN EXIT CODE WITH VALUES..                                     CHE  400
C               0 - OPTIMAL SOLUTION WHICH IS PROBABLY                   CHE  410
C                   NON-UNIQUE (SEE DESCRIPTION).                        CHE  420
C               1 - UNIQUE OPTIMAL SOLUTION.                             CHE  430
C               2 - CALCULATIONS TERMINATED PREMATURELY                  CHE  440
C                   DUE TO ROUNDING ERRORS.                              CHE  450
C IF YOUR FORTRAN COMPILER PERMITS A SINGLE COLUMN OF A TWO              CHE  460
C DIMENSIONAL ARRAY TO BE PASSED TO A ONE DIMENSIONAL ARRAY              CHE  470
C THROUGH A SUBROUTINE CALL, CONSIDERABLE SAVINGS IN                     CHE  480
C EXECUTION TIME MAY BE ACHIEVED THROUGH THE USE OF THE                  CHE  490
C FOLLOWING SUBROUTINE WHICH OPERATES ON COLUMN VECTORS.                 CHE  500
C     SUBROUTINE COL (V1,V2,MLT,NOTROW,I1,NP2)                           CHE  510
C THIS SUBROUTINE SUBTRACTS FROM THE VECTOR V1 A MULTIPLE OF             CHE  520
C THE VECTOR V2 STARTING AT THE I1*TH ELEMENT UP TO THE                  CHE  530
C NP2*TH ELEMENT, EXCEPT FOR THE NOTROW*TH ELEMENT.                      CHE  540
C     REAL V1(NP2),V2(NP2),MLT                                           CHE  550
C     DO 1 I=I1,NP2                                                      CHE  560
```

```
C       IF(I.EQ.NOTROW) GO TO 1                                          CHE  570
C       V1(I)=V1(I)-MLT*V2(I)                                            CHE  580
C   1   CONTINUE                                                         CHE  590
C     RETURN                                                             CHE  600
C     END                                                                CHE  610
C SEE COMMENTS FOLLOWING STATEMENT NUMBER 340 FOR                        CHE  620
C INSTRUCTIONS ON THE IMPLEMENTATION OF THIS MODIFICATION.               CHE  630
      DIMENSION A(NDIM,MDIM), B(MDIM), X(NDIM)                           CHE  640
      INTEGER PROW, PCOL, RANK, RANKP1, OCODE                            CHE  650
C BIG MUST BE SET EQUAL TO ANY VERY LARGE REAL CONSTANT.                 CHE  660
C ITS VALUE HERE IS APPROPRIATE FOR THE IBM/370/145.                     CHE  670
      DATA BIG /1.E+75/                                                  CHE  680
C INITIALIZATION.                                                        CHE  690
      MP1 = M + 1                                                        CHE  700
      NP1 = N + 1                                                        CHE  710
      NP2 = N + 2                                                        CHE  720
      NP3 = N + 3                                                        CHE  730
      NP1MR = 1                                                          CHE  740
      RANK = N                                                           CHE  750
      RELTMP = RELERR                                                    CHE  760
      RELERR = 0.                                                        CHE  770
      DO 10 J=1,M                                                        CHE  780
        A(NP1,J) = 1.                                                    CHE  790
        A(NP2,J) = -B(J)                                                 CHE  800
        A(NP3,J) = N + J                                                 CHE  810
   10 CONTINUE                                                           CHE  820
      A(NP1,MP1) = 0.                                                    CHE  830
      ITER = 0                                                           CHE  840
      OCODE = 1                                                          CHE  850
      DO 20 I=1,N                                                        CHE  860
        X(I) = 0.                                                        CHE  870
        A(I,MP1) = I                                                     CHE  880
   20 CONTINUE                                                           CHE  890
C LEVEL 1.                                                               CHE  900
      LEV = 1                                                            CHE  910
      K = 0                                                              CHE  920
   30 K = K + 1                                                          CHE  930
      KP1 = K + 1                                                        CHE  940
      NP1MK = NP1 - K                                                    CHE  950
      MODE = 0                                                           CHE  960
      DO 40 J=K,M                                                        CHE  970
        B(J) = 1.                                                        CHE  980
   40 CONTINUE                                                           CHE  990
C DETERMINE THE VECTOR TO ENTER THE BASIS.                               CHE 1000
   50 D = -BIG                                                           CHE 1010
      DO 60 J=K,M                                                        CHE 1020
        IF (B(J).EQ.0.) GO TO 60                                         CHE 1030
        DD = ABS(A(NP2,J))                                               CHE 1040
        IF (DD.LE.D) GO TO 60                                            CHE 1050
        PCOL = J                                                         CHE 1060
        D = DD                                                           CHE 1070
   60 CONTINUE                                                           CHE 1080
      IF (K.GT.1) GO TO 70                                               CHE 1090
C TEST FOR ZERO RIGHT-HAND SIDE.                                         CHE 1100
      IF (D.GT.TOL) GO TO 70                                             CHE 1110
      RESMAX = 0.                                                        CHE 1120
      MODE = 2                                                           CHE 1130
      GO TO 380                                                          CHE 1140
C DETERMINE THE VECTOR TO LEAVE THE BASIS.                               CHE 1150
   70 D = TOL                                                            CHE 1160
      DO 80 I=1,NP1MK                                                    CHE 1170
        DD = ABS(A(I,PCOL))                                              CHE 1180
        IF (DD.LE.D) GO TO 80                                            CHE 1190
        PROW = I                                                         CHE 1200
        D = DD                                                           CHE 1210
   80 CONTINUE                                                           CHE 1220
      IF (D.GT.TOL) GO TO 330                                            CHE 1230
C CHECK FOR LINEAR DEPENDENCE IN LEVEL 1.                                CHE 1240
      B(PCOL) = 0.                                                       CHE 1250
      IF (MODE.EQ.1) GO TO 50                                            CHE 1260
      DO 100 J=K,M                                                       CHE 1270
        IF (B(J).EQ.0.) GO TO 100                                        CHE 1280
        DO 90 I=1,NP1MK                                                  CHE 1290
          IF (ABS(A(I,J)).LE.TOL) GO TO 90                               CHE 1300
          MODE = 1                                                       CHE 1310
          GO TO 50                                                       CHE 1320
   90   CONTINUE                                                         CHE 1330
  100 CONTINUE                                                           CHE 1340
      RANK = K - 1                                                       CHE 1350
      NP1MR = NP1 - RANK                                                 CHE 1360
      OCODE = 0                                                          CHE 1370
      GO TO 160                                                          CHE 1380
  110 IF (PCOL.EQ.K) GO TO 130                                           CHE 1390
C INTERCHANGE COLUMNS IN LEVEL 1.                                        CHE 1400
      DO 120 I=1,NP3                                                     CHE 1410
        D = A(I,PCOL)                                                    CHE 1420
        A(I,PCOL) = A(I,K)                                               CHE 1430
        A(I,K) = D                                                       CHE 1440
  120 CONTINUE                                                           CHE 1450
  130 IF (PROW.EQ.NP1MK) GO TO 150                                       CHE 1460
C INTERCHANGE ROWS IN LEVEL 1.                                           CHE 1470
      DO 140 J=1,MP1                                                     CHE 1480
        D = A(PROW,J)                                                    CHE 1490
```

```
            A(PROW,J) = A(NP1MK,J)                                      CHE 1500
            A(NP1MK,J) = D                                              CHE 1510
      140 CONTINUE                                                      CHE 1520
      150 IF (K.LT.N) GO TO 30                                          CHE 1530
      160 IF (RANK.EQ.M) GO TO 380                                      CHE 1540
          RANKP1 = RANK + 1                                             CHE 1550
C LEVEL 2.                                                              CHE 1560
          LEV = 2                                                       CHE 1570
C DETERMINE THE VECTOR TO ENTER THE BASIS                               CHE 1580
          D = TOL                                                       CHE 1590
          DO 170 J=RANKP1,M                                             CHE 1600
            DD = ABS(A(NP2,J))                                          CHE 1610
            IF (DD.LE.D) GO TO 170                                      CHE 1620
            PCOL = J                                                    CHE 1630
            D = DD                                                      CHE 1640
      170 CONTINUE                                                      CHE 1650
C COMPARE CHEBYSHEV ERROR WITH TOL.                                     CHE 1660
          IF (D.GT.TOL) GO TO 180                                       CHE 1670
          RESMAX = 0.                                                   CHE 1680
          MODE = 3                                                      CHE 1690
          GO TO 380                                                     CHE 1700
      180 IF (A(NP2,PCOL).LT.-TOL) GO TO 200                            CHE 1710
          A(NP1,PCOL) = 2. - A(NP1,PCOL)                                CHE 1720
          DO 190 I=NP1MR,NP3                                            CHE 1730
            IF (I.EQ.NP1) GO TO 190                                     CHE 1740
            A(I,PCOL) = -A(I,PCOL)                                      CHE 1750
      190 CONTINUE                                                      CHE 1760
C ARRANGE FOR ALL ENTRIES IN PIVOT COLUMN                               CHE 1770
C (EXCEPT PIVOT) TO BE NEGATIVE.                                        CHE 1780
      200 DO 220 I=NP1MR,N                                              CHE 1790
            IF (A(I,PCOL).LT.TOL) GO TO 220                             CHE 1800
            DO 210 J=1,M                                                CHE 1810
              A(NP1,J) = A(NP1,J) + 2.*A(I,J)                           CHE 1820
              A(I,J) = -A(I,J)                                          CHE 1830
      210     CONTINUE                                                  CHE 1840
            A(I,MP1) = -A(I,MP1)                                        CHE 1850
      220 CONTINUE                                                      CHE 1860
          PROW = NP1                                                    CHE 1870
          GO TO 330                                                     CHE 1880
      230 IF (RANKP1.EQ.M) GO TO 380                                    CHE 1890
          IF (PCOL.EQ.M) GO TO 250                                      CHE 1900
C INTERCHANGE COLUMNS IN LEVEL 2.                                       CHE 1910
          DO 240 I=NP1MR,NP3                                            CHE 1920
            D = A(I,PCOL)                                               CHE 1930
            A(I,PCOL) = A(I,M)                                          CHE 1940
            A(I,M) = D                                                  CHE 1950
      240 CONTINUE                                                      CHE 1960
      250 MM1 = M - 1                                                   CHE 1970
C LEVEL 3.                                                              CHE 1980
          LEV = 3                                                       CHE 1990
C DETERMINE THE VECTOR TO ENTER THE BASIS.                              CHE 2000
      260 D = -TOL                                                      CHE 2010
          VAL = 2.*A(NP2,M)                                             CHE 2020
          DO 280 J=RANKP1,MM1                                           CHE 2030
            IF (A(NP2,J).GE.D) GO TO 270                                CHE 2040
            PCOL = J                                                    CHE 2050
            D = A(NP2,J)                                                CHE 2060
            MODE = 0                                                    CHE 2070
            GO TO 280                                                   CHE 2080
      270   DD = VAL - A(NP2,J)                                         CHE 2090
            IF (DD.GE.D) GO TO 280                                      CHE 2100
            MODE = 1                                                    CHE 2110
            PCOL = J                                                    CHE 2120
            D = DD                                                      CHE 2130
      280 CONTINUE                                                      CHE 2140
          IF (D.GE.-TOL) GO TO 380                                      CHE 2150
          DD = -D/A(NP2,M)                                              CHE 2160
          IF (DD.GE.RELTMP) GO TO 290                                   CHE 2170
          RELERR = DD                                                   CHE 2180
          MODE = 4                                                      CHE 2190
          GO TO 380                                                     CHE 2200
      290 IF (MODE.EQ.0) GO TO 310                                      CHE 2210
          DO 300 I=NP1MR,NP1                                            CHE 2220
            A(I,PCOL) = 2.*A(I,M) - A(I,PCOL)                           CHE 2230
      300 CONTINUE                                                      CHE 2240
          A(NP2,PCOL) = D                                               CHE 2250
          A(NP3,PCOL) = -A(NP3,PCOL)                                    CHE 2260
C DETERMINE THE VECTOR TO LEAVE THE BASIS.                              CHE 2270
      310 D = BIG                                                       CHE 2280
          DO 320 I=NP1MR,NP1                                            CHE 2290
            IF (A(I,PCOL).LE.TOL) GO TO 320                             CHE 2300
            DD = A(I,M)/A(I,PCOL)                                       CHE 2310
            IF (DD.GE.D) GO TO 320                                      CHE 2320
            PROW = I                                                    CHE 2330
            D = DD                                                      CHE 2340
      320 CONTINUE                                                      CHE 2350
          IF (D.LT.BIG) GO TO 330                                       CHE 2360
          OCODE = 2                                                     CHE 2370
          GO TO 380                                                     CHE 2380
C PIVOT ON A(PROW,PCOL).                                                CHE 2390
      330 PIVOT = A(PROW,PCOL)                                          CHE 2400
          DO 340 J=1,M                                                  CHE 2410
            A(PROW,J) = A(PROW,J)/PIVOT                                 CHE 2420
```

```
  340 CONTINUE                                                          CHE 2430
C IF PERMITTED, USE SUBROUTINE COL IN THE DESCRIPTION                   CHE 2440
C SECTION AND REPLACE THE FOLLOWING EIGHT STATEMENTS DOWN TO            CHE 2450
C AND INCLUDING STATEMENT NUMBER 360 BY..                               CHE 2460
C     DO 360 J=1,M                                                      CHE 2470
C       IF(J.EQ.PCOL) GO TO 360                                         CHE 2480
C       CALL COL (A(1,J),A(1,PCOL),A(PROW,J),PROW,NP1MR,NP2)            CHE 2490
C 360 CONTINUE                                                          CHE 2500
      DO 360 J=1,M                                                      CHE 2510
        IF (J.EQ.PCOL) GO TO 360                                        CHE 2520
        D = A(PROW,J)                                                   CHE 2530
        DO 350 I=NP1MR,NP2                                              CHE 2540
          IF (I.EQ.PROW) GO TO 350                                      CHE 2550
          A(I,J) = A(I,J) - D*A(I,PCOL)                                 CHE 2560
  350   CONTINUE                                                        CHE 2570
  360 CONTINUE                                                          CHE 2580
      TPIVOT = -PIVOT                                                   CHE 2590
      DO 370 I=NP1MR,NP2                                                CHE 2600
        A(I,PCOL) = A(I,PCOL)/TPIVOT                                    CHE 2610
  370 CONTINUE                                                          CHE 2620
      A(PROW,PCOL) = 1./PIVOT                                           CHE 2630
      D = A(PROW,MP1)                                                   CHE 2640
      A(PROW,MP1) = A(NP3,PCOL)                                         CHE 2650
      A(NP3,PCOL) = D                                                   CHE 2660
      ITER = ITER + 1                                                   CHE 2670
      GO TO (110, 230, 260), LEV                                        CHE 2680
C PREPARE OUTPUT.                                                       CHE 2690
  380 DO 390 J=1,M                                                      CHE 2700
        B(J) = 0.                                                       CHE 2710
  390 CONTINUE                                                          CHE 2720
      IF (MODE.EQ.2) GO TO 450                                          CHE 2730
      DO 400 J=1,RANK                                                   CHE 2740
        K = A(NP3,J)                                                    CHE 2750
        X(K) = A(NP2,J)                                                 CHE 2760
  400 CONTINUE                                                          CHE 2770
      IF (MODE.EQ.3 .OR. RANK.EQ.M) GO TO 450                           CHE 2780
      DO 410 I=NP1MR,NP1                                                CHE 2790
        K = ABS(A(I,MP1)) - FLOAT(N)                                    CHE 2800
        B(K) = A(NP2,M)*SIGN(1.,A(I,MP1))                               CHE 2810
  410 CONTINUE                                                          CHE 2820
      IF (RANKP1.EQ.M) GO TO 430                                        CHE 2830
      DO 420 J=RANKP1,MM1                                               CHE 2840
        K = ABS(A(NP3,J)) - FLOAT(N)                                    CHE 2850
        B(K) = (A(NP2,M)-A(NP2,J))*SIGN(1.,A(NP3,J))                    CHE 2860
  420 CONTINUE                                                          CHE 2870
C TEST FOR NON-UNIQUE SOLUTION.                                         CHE 2880
  430 DO 440 I=NP1MR,NP1                                                CHE 2890
        IF (ABS(A(I,M)).GT.TOL) GO TO 440                               CHE 2900
        OCODE = 0                                                       CHE 2910
        GO TO 450                                                       CHE 2920
  440 CONTINUE                                                          CHE 2930
  450 IF (MODE.NE.2 .AND. MODE.NE.3) RESMAX = A(NP2,M)                  CHE 2940
      IF (RANK.EQ.M) RESMAX = 0.                                        CHE 2950
      IF (MODE.EQ.4) RESMAX = RESMAX - D                                CHE 2960
      RETURN                                                            CHE 2970
      END                                                               CHE 2980
```

ALGORITHM 496
The LZ Algorithm To Solve the Generalized Eigenvalue Problem for Complex Matrices [F2]

LINDA KAUFMAN

University of Colorado

Key Words and Phrases: eigenvalues, generalized eigenvalue problem
CR Categories: 5.14
Language: Fortran

DESCRIPTION

The program given here is an implementation of the LZ algorithm [1] for finding $\mathbf{x}$ and λ such that $A\mathbf{x} = \lambda B\mathbf{x}$, where A and B are $n \times n$ matrices. The LZ algorithm does not require matrix inversion and may be used when either or both matrices are singular. It is a generalization of Rutishauser's LR method [3] for the standard eigenvalue problem $A\mathbf{x} = \lambda\mathbf{x}$ and closely resembles the QZ algorithm given by Moler and Stewart [2] for the generalized problem given above. Since the LZ algorithm uses elementary transformations and the QZ algorithm uses orthogonal transformations, the LZ algorithm is probably more efficient when either A or B is complex.

The LZ algorithm is based on three observations:

(1) If L and M are nonsingular matrices, the eigenvalue problems $LAM\mathbf{y} = \lambda LBM\mathbf{y}$ and $A\mathbf{x} = \lambda B\mathbf{x}$ have the same eigenvalues and their eigenvectors are related by $\mathbf{x} = M\mathbf{y}$.

(2) If A is a triangular matrix with diagonal elements α_i, and B is a triangular matrix with diagonal elements β_i, then for each i, $i = 1, 2, \ldots, n$, α_i/β_i is an eigenvalue of the generalized problem if $\beta_i \neq 0$.

(3) There exist matrices L and M such that LAM and LBM are upper triangular and L and M are the products of lower triangular and permutation matrices.

The aim of the LZ algorithm is to find the matrices L and M which reduce A and B simultaneously to triangular form. It accomplishes this aim in two phases.

(1) The first phase simultaneously reduces A to upper Hessenberg form (i.e. $a_{i,j} = 0$ for $i > j$) and B to upper triangular form by (a) reducing B to triangular form as in Gaussian elimination and applying the transformations to A, and (b) reducing A to upper Hessenberg form while preserving the triangularity of B by using row transformations to zero the elements below the subdiagonal of A and column transformations to zero the subdiagonal elements of B introduced by the row transformations.

(2) The second phase iteratively reduces A to upper triangular form while preserving the triangularity of B where each iteration is given by (a) finding a shift λ_k, (b) finding a stabilized transformation $\bar{L}_k$ such that $\bar{L}_k(A_k - \lambda_k B_k)$ is

Received 21 June 1973, 6 December 1974, and 15 April 1975.

Author's address: Department of Computer Science, University of Colorado, Boulder, CO 80302.

upper triangular, (c) finding a stabilized transformation $\bar{M}_k$ such that $\bar{L}_k B_k \bar{M}_k$ is upper triangular, (d) setting $A_{k+1} = \bar{L}_k A_k \bar{M}_k$ and $B_{k+1} = \bar{L}_k B_k \bar{M}_k$.

If $\bar{L}_k$ and $\bar{M}_k$ are constructed appropriately, A_{k+1} will be upper Hessenberg and hopefully some of its subdiagonal elements will be small enough to be considered zero.

The shift λ_k is used to hasten the convergence of the algorithm. In practice it is set to the solution of the lowest 2×2 subproblem on the diagonal of $A_k - \lambda B_k$ which has not been triangularized.

Each matrix $\bar{L}_k$ is a product of matrices $L_{k,n-1} L_{k,n-2} \ldots L_{k,2} L_{k,1}$ and each $\bar{M}_k$ is a product of matrices $M_{k,1} M_{k,2} \ldots M_{k,n-1}$, where each $L_{k,i}$ and each $M_{k,i}$ is a stabilized elementary transformation (see [4]). The matrix $L_{k,1}$ is designed to annihilate the (2, 1) element of $A_k - \lambda_k B_k$. For $i \geq 1$, $M_{k,i}$ zeros the ith subdiagonal element of B introduced by $L_{k,i}$, and for $i > 1$, $L_{k,i}$ zeros the $(i + 1, i - 1)$-th element of A introduced by $M_{k,i-1}$.

The LZ algorithm determines the eigenvectors of the problem by computing the eigenvectors of the triangularized A and B and multiplying them by the product of all the column transformations.

The Fortran program given here is designed to find the eigensystem for two complex matrices A and B and consists of two subroutines which must be called separately. A user should first issue a call to LZHES to reduce A to upper Hessenberg form and B to triangular form and then issue a call to LZHES to determine the eigensystem of the reduced matrices. If A and B have already been reduced to Hessenberg and triangular form, then calling LZHES is unnecessary. In this situation, if eigenvectors are requested, the matrix X should be set to the identity matrix. The subroutine LZIT does not return the eigenvalues of the problem but the diagonal elements of the triangularized matrices. The ith eigenvalue may be found by dividing EIGA(i) by EIGB(i). The user is warned that EIGB(i) may be zero. The eigenvectors produced by LZIT are normalized so that the modulus of their largest component is 1.

Entrance to LZHES is attained by the statement

```
CALL LZHES(N,A,NA,B,NB,X,NX,WANTX)
```

where the input parameters are

N	order of the A and B matrices,
A	$n \times n$ complex matrix,
NA	row dimension of A,
B	$n \times n$ complex matrix,
NB	row dimension of B,
NX	row dimension of X (see output parameters)
WANTX	logical variable which should be set to .TRUE. if eigenvectors are wanted and otherwise set to .FALSE.,

and the output parameters are

A	complex upper Hessenberg matrix; the original A matrix is destroyed,
B	complex upper triangular matrix; the original B matrix is destroyed,
X	$n \times n$ complex array containing the transformations needed to compute the eigenvectors of the problem if WANTX was .TRUE..

Entrance to LZIT is attained by the statement

```
CALL LZIT(N,A,NA,B,NB,X,NX,WANTX,ITER,EIGA,EIGB)
```

where the input parameters are

N	order of the A and B matrices,
A	$n \times n$ complex upper Hessenberg matrix,
NA	row dimension of the A matrix,
B	$n \times n$ complex upper triangular matrix,
NB	row dimension of the B matrix,
X	$n \times n$ complex array containing the transformations to determine the

eigenvectors of the original problem if WANTX has been set to .TRUE. ,

NX row dimension of the X matrix,

WANTX logical variable which should be set to .TRUE. if eigenvectors are wanted and otherwise set to .FALSE. ,

and the output parameters are

X $n \times n$ complex array whose ith column contains the ith eigenvectors if WANTX was set to .TRUE. ,

ITER integer array of length n whose ith entry contains the number of iterations needed to find the ith eigenvalue; for any i, if iter$(i) = -1$, then after 30 iterations there was not a sufficient decrease in the last subdiagonal element of A to continue iterating so that not all the eigenvalues have been isolated,

EIGA complex array of length n containing the diagonal of the triangularized A,

EIGB complex array of length n containing the diagonal of B.

Since $| x |$, where x is a complex scalar, must be computed at least $8n$ times per iteration, the execution time of the method is dependent on the algorithm used to compute $| x |$. Replacing the true modulus by an approximation has little effect on the numerical properties but may result in a significantly more efficient code. Several versions of the LZ algorithm with different methods for computing $| x |$ were run on the IBM 360 model 67 with the Fortran H compiler opt=2. The results were interesting:

(1) The procedure CABS supplied by the manufacturer to compute $| x |$ was by far the slowest.

```
 THE  MATRIX A:

  -238+  -344I     86+   178I    164+   240I   -166+  -308I     56+   158I
    76+   152I    -96+  -128I     40+   -32I     60+   184I    -60+  -136I
   118+   284I     55+  -182I    -13+   460I     34+  -192I   -176+  -214I
  -314+  -160I    132+    78I    114+   296I    -90+  -164I   -424+  -374I
   -54+   -24I   -205+  -400I    109+   148I    158+   312I    -38+   -96I

THE MATRIX B:

   388+    94I   -386+  -122I   -250+   -14I    556+   130I   -396+   -62I
  -304+   -76I    384+    64I   -160+    16I   -240+   -92I    240+    68I
  -658+  -136I    -73+   100I   -109+  -250I   -118+   100I    406+    96I
  -640+   -10I    204+   -42I   -692+   -90I    288+    66I   -192+   154I
  -162+   -72I    631+   158I    131+    52I   -758+  -184I    278+    76I

   TRUE EIGENVALUE
  1     7.64705882352 94E-01+     9.41176470588 23E-01 I
  2    -1.00000000000 00E+00+    -1.33333333333 33E+00 I
  3    -3.52941176470 59E-01+    -4.11764705882 35E-01 I
  4    -3.52941176470 59E-01+    -4.11764705882 35E-01 I
  5    -3.52941176470 59E-01+     4.11764705882 35E-01 I
ALPHA
  1    -3.10592780609 29E+02+    -1.23809677987 94E+03 I
  2     3.94321130207 88E+01+     1.85520489460 44E+02 I
  3     3.43035787177 87E+02+     3.62670034078 91E+01 I
  4     4.03319231953 21E+01+    -4.14757682755 21E+02 I
  5    -2.64839041497 05E+01+    -1.23832486793 01E+02 I
BETA
  1    -9.53890185039 61E+02+    -4.45030945947 35E+02 I
  2    -1.03245395628 50E+02+    -4.78599619557 82E+01 I
  3    -4.62416749384 52E+02+     4.36729697959 56E+02 I
  4     5.32262448022 92E+02+     5.54173911779 70E+02 I
  5    -1.41584796530 56E+02+     1.85676449961 21E+02 I
   COMPUTED EIGENVALUE
  1     7.64705882352 96E-01+     9.41176470588 21E-01 I
  2    -1.00000000000 00E+00+    -1.33333333333 33E+00 I
  3    -3.52941176470 59E-01+    -4.11764705882 33E-01 I
  4    -3.52941176470 58E-01+    -4.11764705882 36E-01 I
  5    -3.52941176470 58E-01+     4.11764705882 32E-01 I
          RELATIVE ERROR      RELATIVE RESIDUAL      NO. OF ITERATIONS
  1     2.70099715873 30E-14     4.31385892763 75E-15        0
  2     6.02915504134 57E-15     6.78869273623 43E-15        1
  3     3.94415226011 35E-14     2.97426304528 89E-15        0
  4     1.18097674847 17E-14     6.00575930102 25E-15        3
  5     5.65428640650 59E-14     8.30808706925 88E-15        1
```

Fig. 1

(2) Substituting the procedure RABS given by

```
REAL FUNCTION RABS(X)
COMPLEX X, XX
REAL T(2)
EQUIVALENCE (XX, T(1))
XX = X
RABS = ABS(T(1)) + ABS(T(2))
RETURN
END
```

for CABS decreased the time by about 20 percent when no eigenvectors were computed. When using RABS, changing the precision of the program requires very little effort.

(3) Computing $|x|$ as ABS(REAL(X)) + ABS(AIMAG(X)) as in the program given here decreased execution times from version 2 by roughly 10 percent when no eigenvectors were computed.

Although theoretically numerical growth can be quite large, our experiments have shown that the algorithm behaves like most other methods which use stabilized elementary transformations: numerical growth rarely occurs and errors rarely accumulate. The eigenvalues are usually determined to the accuracy justified by the condition of the problem.

The example shown in Figure 1 was generated in integer arithmetic by multiplying two diagonal matrices by nonsingular transformations. It was run on the CDC Run compiler. The relative residual is the quantity,

$$|| \beta_i A \mathbf{x}_i - \alpha_i B \mathbf{x}_i ||_\infty / (|| \beta_i || \, || A ||_\infty + || \alpha_i || \, || B ||_\infty)$$

where α_i and β_i are the ith diagnonal elements of the triangularized A and B matrices and $\mathbf{x}_i$ is the ith eigenvector.

REFERENCES

1. KAUFMAN, L.C. The LZ algorithm to solve the generalized eigenvalue problem. *SIAM J. Numer. Anal. 11* (Oct. 1974), 997–1023.
2. MOLER, C.B., AND STEWART, G.W. An algorithm for the generalized matrix eigenvalue problem. *SIAM J. Numer. Anal. 10* (April 1973), 241–256.
3. RUTISHAUSER, H. *Solution of the eigenvalue problem with the LR transformation.* Nat. Bur. Standards Appl. Math. Ser. 49, U.S. Govt. Printing Office, Washington, D.C., Jan. 1958.
4. WILKINSON, J.H. *The Algebraic Eigenvalue Problem.* Oxford U. Press, New York, 1965.

ALGORITHM

```
      SUBROUTINE LZHES(N, A, NA, B, NB, X, NX, WANTX)                    LZH   10
C THIS SUBROUTINE REDUCES THE COMPLEX MATRIX A TO UPPER                   LZH   20
C HESSENBERG FORM AND REDUCES THE COMPLEX MATRIX B TO                     LZH   30
C TRIANGULAR FORM                                                         LZH   40
C INPUT PARAMETERS..                                                      LZH   50
C N   THE ORDER OF THE A AND B MATRICES                                   LZH   60
C A   A COMPLEX MATRIX                                                    LZH   70
C NA  THE ROW DIMENSION OF THE A MATRIX                                   LZH   80
C B   A COMPLEX MATRIX                                                    LZH   90
C NB  THE ROW DIMENSION OF THE B MATRIX                                   LZH  100
C NX  THE ROW DIMENSION OF THE X MATRIX                                   LZH  110
C WANTX A LOGICAL VARIABLE WHICH IS SET TO .TRUE. IF                      LZH  120
C       THE EIGENVECTORS ARE WANTED. OTHERWISE IT SHOULD                  LZH  130
C     BE SET TO .FALSE.                                                   LZH  140
C OUTPUT PARAMETERS..                                                     LZH  150
C A  ON OUTPUT A IS AN UPPER HESSENBERG MATRIX, THE                       LZH  160
C    ORIGINAL MATRIX HAS BEEN DESTROYED                                   LZH  170
C B  AN UPPER TRIANGULAR MATRIX, THE ORIGINAL MATRIX                      LZH  180
C    HAS BEEN DESTROYED                                                   LZH  190
C X  CONTAINS THE TRANSFORMATIONS NEEDED TO COMPUTE                       LZH  200
C    THE EIGENVECTORS OF THE ORIGINAL SYSTEM                              LZH  210
      COMPLEX Y, W, Z, A(NA,N), B(NB,N), X(NX,N)                          LZH  220
      REAL C, D                                                           LZH  230
      LOGICAL WANTX                                                       LZH  240
      NM1 = N - 1                                                         LZH  250
C REDUCE B TO TRIANGULAR FORM USING ELEMENTARY                            LZH  260
C TRANSFORMATIONS                                                         LZH  270
      DO 80 I=1,NM1                                                       LZH  280
        D = 0.00                                                          LZH  290
        IP1 = I + 1                                                       LZH  300
```

```
      DO 10 K=IP1,N                                                    LZH  310
        Y = B(K,I)                                                     LZH  320
        C = ABS(REAL(Y)) + ABS(AIMAG(Y))                               LZH  330
        IF (C.LE.D) GO TO 10                                           LZH  340
        D = C                                                          LZH  350
        II = K                                                         LZH  360
 10   CONTINUE                                                         LZH  370
      IF (D.EQ.0.0) GO TO 80                                           LZH  380
      Y = B(I,1)                                                       LZH  390
      IF (D.LE.ABS(REAL(Y))+ABS(AIMAG(Y))) GO TO 40                    LZH  400
C MUST INTERCHANGE                                                     LZH  410
      DO 20 J=1,N                                                      LZH  420
        Y = A(I,J)                                                     LZH  430
        A(I,J) = A(II,J)                                               LZH  440
        A(II,J) = Y                                                    LZH  450
 20   CONTINUE                                                         LZH  460
      DO 30 J=I,N                                                      LZH  470
        Y = B(I,J)                                                     LZH  480
        B(I,J) = B(II,J)                                               LZH  490
        B(II,J) = Y                                                    LZH  500
 30   CONTINUE                                                         LZH  510
 40   DO 70 J=IP1,N                                                    LZH  520
        Y = B(J,I)/B(I,I)                                              LZH  530
        IF (REAL(Y).EQ.0.0 .AND. AIMAG(Y).EQ.0.0) GO TO 70             LZH  540
        DO 50 K=1,N                                                    LZH  550
          A(J,K) = A(J,K) - Y*A(I,K)                                   LZH  560
 50     CONTINUE                                                       LZH  570
        DO 60 K=IP1,N                                                  LZH  580
          B(J,K) = B(J,K) - Y*B(I,K)                                   LZH  590
 60     CONTINUE                                                       LZH  600
 70   CONTINUE                                                         LZH  610
      B(IP1,I) = (0.0,0.0)                                             LZH  620
 80 CONTINUE                                                           LZH  630
C INITIALIZE X                                                         LZH  640
      IF (.NOT.WANTX) GO TO 110                                        LZH  650
      DO 100 I=1,N                                                     LZH  660
        DO 90 J=1,N                                                    LZH  670
          X(I,J) = (0.0,0.0)                                           LZH  680
 90     CONTINUE                                                       LZH  690
        X(I,I) = (1.0,0.00)                                            LZH  700
100 CONTINUE                                                           LZH  710
C REDUCE A TO UPPER HESSENBERG FORM                                    LZH  720
110 NM2 = N - 2                                                        LZH  730
      IF (NM2.LT.1) GO TO 270                                          LZH  740
      DO 260 J=1,NM2                                                   LZH  750
        JM2 = NM1 - J                                                  LZH  760
        JP1 = J + 1                                                    LZH  770
        DO 250 II=1,JM2                                                LZH  780
          I = N + 1 - II                                               LZH  790
          IM1 = I - 1                                                  LZH  800
          IMJ = I - J                                                  LZH  810
          W = A(I,J)                                                   LZH  820
          Z = A(IM1,J)                                                 LZH  830
          IF (ABS(REAL(W))+ABS(AIMAG(W)).LE.ABS(REAL(Z))               LZH  840
     *     +ABS(AIMAG(Z))) GO TO 140                                   LZH  850
C MUST INTERCHANGE ROWS                                                LZH  860
          DO 120 K=J,N                                                 LZH  870
            Y = A(I,K)                                                 LZH  880
            A(I,K) = A(IM1,K)                                          LZH  890
            A(IM1,K) = Y                                               LZH  900
120       CONTINUE                                                     LZH  910
          DO 130 K=IM1,N                                               LZH  920
            Y = B(I,K)                                                 LZH  930
            B(I,K) = B(IM1,K)                                          LZH  940
            B(IM1,K) = Y                                               LZH  950
130       CONTINUE                                                     LZH  960
140       Z = A(I,J)                                                   LZH  970
          IF (REAL(Z).EQ.0.0 .AND. AIMAG(Z).EQ.0.0) GO TO 170          LZH  980
          Y = Z/A(IM1,J)                                               LZH  990
          DO 150 K=JP1,N                                               LZH 1000
            A(I,K) = A(I,K) - Y*A(IM1,K)                               LZH 1010
150       CONTINUE                                                     LZH 1020
          DO 160 K=IM1,N                                               LZH 1030
            B(I,K) = B(I,K) - Y*B(IM1,K)                               LZH 1040
160       CONTINUE                                                     LZH 1050
C TRANSFORMATION FROM THE RIGHT                                        LZH 1060
170       W = B(I,IM1)                                                 LZH 1070
          Z = B(I,I)                                                   LZH 1080
          IF (ABS(REAL(W))+ABS(AIMAG(W)).LE.ABS(REAL(Z))               LZH 1090
     *     +ABS(AIMAG(Z))) GO TO 210                                   LZH 1100
C MUST INTERCHANGE COLUMNS                                             LZH 1110
          DO 180 K=1,I                                                 LZH 1120
            Y = B(K,I)                                                 LZH 1130
            B(K,I) = B(K,IM1)                                          LZH 1140
            B(K,IM1) = Y                                               LZH 1150
180       CONTINUE                                                     LZH 1160
          DO 190 K=1,N                                                 LZH 1170
            Y = A(K,I)                                                 LZH 1180
            A(K,I) = A(K,IM1)                                          LZH 1190
            A(K,IM1) = Y                                               LZH 1200
190       CONTINUE                                                     LZH 1210
          IF (.NOT.WANTX) GO TO 210                                    LZH 1220
```

```
            DO 200 K=IMJ,N                                                  LZH 1230
              Y = X(K,I)                                                    LZH 1240
              X(K,I) = X(K,IM1)                                             LZH 1250
              X(K,IM1) = Y                                                  LZH 1260
  200       CONTINUE                                                        LZH 1270
  210       Z = B(I,IM1)                                                    LZH 1280
            IF (REAL(Z).EQ.0.0 .AND. AIMAG(Z).EQ.0.0) GO TO 250             LZH 1290
            Y = Z/B(I,I)                                                    LZH 1300
            DO 220 K=1.IM1                                                  LZH 1310
              B(K,IM1) = B(K,IM1) - Y*B(K,I)                                LZH 1320
  220       CONTINUE                                                        LZH 1330
            B(I,IM1) = (0.0,0.0)                                            LZH 1340
            DO 230 K=1,N                                                    LZH 1350
              A(K,IM1) = A(K,IM1) - Y*A(K,I)                                LZH 1360
  230       CONTINUE                                                        LZH 1370
            IF (.NOT.WANTX) GO TO 250                                       LZH 1380
            DO 240 K=IMJ,N                                                  LZH 1390
              X(K,IM1) = X(K,IM1) - Y*X(K,I)                                LZH 1400
  240       CONTINUE                                                        LZH 1410
  250     CONTINUE                                                          LZH 1420
          A(JP1+1,J) = (0.0,0.0)                                            LZH 1430
  260 CONTINUE                                                              LZH 1440
  270 RETURN                                                                LZH 1450
      END                                                                   LZH 1460


      SUBROUTINE LZIT(N, A, NA, B, NB, X, NX, WANTX, ITER, EIGA,            LZI   10
     * EIGB)                                                                LZI   20
C THIS SUBROUTINE SOLVES THE GENERALIZED EIGENVALUE PROBLEM                 LZI   30
C               A X  = LAMBDA B X                                           LZI   40
C WHERE A IS A COMPLEX UPPER HESSENBERG MATRIX OF                           LZI   50
C ORDER N AND B IS A COMPLEX UPPER TRIANGULAR MATRIX OF ORDER N              LZI   60
C INPUT PARAMETERS                                                          LZI   70
C N       ORDER OF A AND B                                                  LZI   80
C A       AN N X N UPPER HESSENBERG COMPLEX MATRIX                          LZI   90
C NA      THE ROW DIMENSION OF THE A MATRIX                                 LZI  100
C B       AN N X N UPPER TRIANGULAR COMPLEX MATRIX                          LZI  110
C NB      THE ROW DIMENSION OF THE B MATRIX                                 LZI  120
C X       CONTAINS TRANSFORMATIONS TO OBTAIN EIGENVECTORS OF                LZI  130
C         ORIGINAL SYSTEM. IF EIGENVECTORS ARE REQUESTED AND QZHES          LZI  140
C         IS NOT CALLED, X SHOULD BE SET TO THE IDENTITY MATRIX             LZI  150
C NX      THE ROW DIMENSION OF THE X MATRIX                                 LZI  160
C WANTX   LOGICAL VARIABLE WHICH SHOULD BE SET TO .TRUE.                    LZI  170
C         IF EIGENVECTORS ARE WANTED. OTHERWISE IT                          LZI  180
C         SHOULD BE SET TO .FALSE.                                          LZI  190
C OUTPUT PARAMETERS                                                         LZI  200
C X       THE ITH COLUMN CONTAINS THE ITH EIGENVECTOR                       LZI  210
C         IF EIGENVECTORS ARE REQUESTED                                     LZI  220
C ITER    AN INTEGER ARRAY OF LENGTH N WHOSE ITH ENTRY                      LZI  230
C         CONTAINS THE NUMBER OF ITERATIONS NEEDED TO FIND                  LZI  240
C         THE ITH EIGENVALUE. FOR ANY I IF ITER(I) =-1 THEN                 LZI  250
C         AFTER 30 ITERATIONS THERE HAS NOT BEEN A SUFFICIENT               LZI  260
C         DECREASE IN THE LAST SUBDIAGONAL ELEMENT OF A                     LZI  270
C         TO CONTINUE ITERATING.                                            LZI  280
C EIGA    A COMPLEX ARRAY OF LENGTH N CONTAINING THE DIAGONAL OF A          LZI  290
C EIGB    A COMPLEX ARRAY OF LENGTH N CONTAINING THE DIAGONAL OF B          LZI  300
C THE ITH EIGENVALUE CAN BE FOUND BY DIVIDING EIGA(I) BY                    LZI  310
C EIGB(I). WATCH OUT FOR EIGB(I) BEING ZERO                                 LZI  320
      COMPLEX A(NA,N), B(NB,N), EIGA(N), EIGB(N)                            LZI  330
      COMPLEX S, W, Y, Z, CSQRT                                             LZI  340
      COMPLEX X(NX,N)                                                       LZI  350
      INTEGER ITER(N)                                                       LZI  360
      COMPLEX ANNM1, ALFM, BETM, D, SL, DEN, NUM, ANM1M1                    LZI  370
      REAL EPSA, EPSB, SS, R, ANORM, BNORM, ANI, BNI, C                     LZI  380
      REAL D0, D1, D2, E0, E1                                               LZI  390
      LOGICAL WANTX                                                         LZI  400
      NN = N                                                                LZI  410
C COMPUTE THE MACHINE PRECISION TIMES THE NORM OF A AND B                   LZI  420
      ANORM = 0.                                                            LZI  430
      BNORM = 0.                                                            LZI  440
      DO 30 I=1,N                                                           LZI  450
        ANI = 0.                                                            LZI  460
        IF (I.EQ.1) GO TO 10                                                LZI  470
        Y = A(I,I-1)                                                        LZI  480
        ANI = ANI + ABS(REAL(Y)) + ABS(AIMAG(Y))                            LZI  490
   10   BNI = 0.                                                            LZI  500
        DO 20 J=I,N                                                         LZI  510
          ANI = ANI + ABS(REAL(A(I,J))) + ABS(AIMAG(A(I,J)))                LZI  520
          BNI = BNI + ABS(REAL(B(I,J))) + ABS(AIMAG(B(I,J)))                LZI  530
   20   CONTINUE                                                            LZI  540
        IF (ANI.GT.ANORM) ANORM = ANI                                       LZI  550
        IF (BNI.GT.BNORM) BNORM = BNI                                       LZI  560
   30 CONTINUE                                                              LZI  570
      IF (ANORM.EQ.0.) ANORM = 1.0                                          LZI  580
      IF (BNORM.EQ.0.) BNORM = 1.0                                          LZI  590
      EPSB = BNORM                                                          LZI  600
      EPSA = ANORM                                                          LZI  610
   40 EPSA = EPSA/2.0                                                       LZI  620
      EPSB = EPSB/2.0                                                       LZI  630
      C = ANORM + EPSA                                                      LZI  640
      IF (C.GT.ANORM) GO TO 40                                              LZI  650
      IF (N.LE.1) GO TO 320                                                 LZI  660
```

```
   50 ITS = 0                                                          LZI  670
      NM1 = NN - 1                                                     LZI  680
C CHECK FOR NEGLIGIBLE SUBDIAGONAL ELEMENTS                            LZI  690
   60 D2 = ABS(REAL(A(NN,NN))) + ABS(AIMAG(A(NN,NN)))                  LZI  700
      DO 70 LB=2,NN                                                    LZI  710
        L = NN + 2 - LB                                                LZI  720
        SS = D2                                                        LZI  730
        Y = A(L-1,L-1)                                                 LZI  740
        D2 = ABS(REAL(Y)) + ABS(AIMAG(Y))                              LZI  750
        SS = SS + D2                                                   LZI  760
        Y = A(L,L-1)                                                   LZI  770
        R = SS + ABS(REAL(Y)) + ABS(AIMAG(Y))                          LZI  780
        IF (R.EQ.SS) GO TO 80                                          LZI  790
   70 CONTINUE                                                         LZI  800
      L = 1                                                            LZI  810
   80 IF (L.EQ.NN) GO TO 320                                           LZI  820
      IF (ITS.LT.30) GO TO 90                                          LZI  830
      ITER(NN) = -1                                                    LZI  840
      IF (ABS(REAL(A(NN,NM1)))+ABS(AIMAG(A(NN,NM1))).GT.0.8*           LZI  850
     * ABS(REAL(ANNM1))+ABS(AIMAG(ANNM1))) RETURN                      LZI  860
   90 IF (ITS.EQ.10 .OR. ITS.EQ.20) GO TO 110                          LZI  870
C COMPUTE SHIFT AS EIGENVALUE OF LOWER 2 BY 2                          LZI  880
      ANNM1 = A(NN,NM1)                                                LZI  890
      ANM1M1 = A(NM1,NM1)                                              LZI  900
      S = A(NN,NN)*B(NM1,NM1) - ANNM1*B(NM1,NN)                        LZI  910
      W = ANNM1*B(NN,NN)*(A(NM1,NN)*B(NM1,NM1)-B(NM1,NN)*ANM1M1)       LZI  920
      Y = (ANM1M1*B(NN,NN)-S)/2.                                       LZI  930
      Z = CSQRT(Y*Y+W)                                                 LZI  940
      IF (REAL(Z).EQ.0.0 .AND. AIMAG(Z).EQ.0.0) GO TO 100              LZI  950
      D0 = REAL(Y/Z)                                                   LZI  960
      IF (D0.LT.0.0) Z = -Z                                            LZI  970
  100 DEN = (Y+Z)*B(NM1,NM1)*B(NN,NN)                                  LZI  980
      IF (REAL(DEN).EQ.0.0 .AND. AIMAG(DEN).EQ.0.0) DEN =              LZI  990
     * CMPLX(EPSA,0.0)                                                 LZI 1000
      NUM = (Y+Z)*S - W                                                LZI 1010
      GO TO 120                                                        LZI 1020
C AD-HOC SHIFT                                                         LZI 1030
  110 Y = A(NM1,NN-2)                                                  LZI 1040
      NUM = CMPLX(ABS(REAL(ANNM1))+ABS(AIMAG(ANNM1)),ABS(REAL(Y)))     LZI 1050
     * +ABS(AIMAG(Y)))                                                 LZI 1060
      DEN = (1.0,0.0)                                                  LZI 1070
C CHECK FOR 2 CONSECUTIVE SMALL SUBDIAGONAL ELEMENTS                   LZI 1080
  120 IF (NN.EQ.L+1) GO TO 140                                         LZI 1090
      D2 = ABS(REAL(A(NM1,NM1))) + ABS(AIMAG(A(NM1,NM1)))              LZI 1100
      E1 = ABS(REAL(ANNM1)) + ABS(AIMAG(ANNM1))                        LZI 1110
      D1 = ABS(REAL(A(NN,NN))) + ABS(AIMAG(A(NN,NN)))                  LZI 1120
      NL = NN - (L+1)                                                  LZI 1130
      DO 130 MB=1,NL                                                   LZI 1140
        M = NN - MB                                                    LZI 1150
        E0 = E1                                                        LZI 1160
        Y = A(M,M-1)                                                   LZI 1170
        E1 = ABS(REAL(Y)) + ABS(AIMAG(Y))                              LZI 1180
        D0 = D1                                                        LZI 1190
        D1 = D2                                                        LZI 1200
        Y = A(M-1,M-1)                                                 LZI 1210
        D2 = ABS(REAL(Y)) + ABS(AIMAG(Y))                              LZI 1220
        Y = A(M,M)*DEN - B(M,M)*NUM                                    LZI 1230
        D0 = (D0+D1+D2)*(ABS(REAL(Y))+ABS(AIMAG(Y)))                   LZI 1240
        E0 = E0*E1*(ABS(REAL(DEN))+ABS(AIMAG(DEN))) + D0               LZI 1250
        IF (E0.EQ.D0) GO TO 150                                        LZI 1260
  130 CONTINUE                                                         LZI 1270
  140 M = L                                                            LZI 1280
  150 CONTINUE                                                         LZI 1290
      ITS = ITS + 1                                                    LZI 1300
      W = A(M,M)*DEN - B(M,M)*NUM                                      LZI 1310
      Z = A(M+1,M)*DEN                                                 LZI 1320
      D1 = ABS(REAL(Z)) + ABS(AIMAG(Z))                                LZI 1330
      D2 = ABS(REAL(W)) + ABS(AIMAG(W))                                LZI 1340
C FIND L AND M AND SET A=LAM AND B=LBM                                 LZI 1350
      NP1 = N + 1                                                      LZI 1360
      LOR1 = L                                                         LZI 1370
      NNORN = NN                                                       LZI 1380
      IF (.NOT.WANTX) GO TO 160                                        LZI 1390
      LOR1 = 1                                                         LZI 1400
      NNORN = N                                                        LZI 1410
  160 DO 310 I=M,NM1                                                   LZI 1420
        J = I + 1                                                      LZI 1430
C FIND ROW TRANSFORMATIONS TO RESTORE A TO                             LZI 1440
C UPPER HESSENBERG FORM. APPLY TRANSFORMATIONS                         LZI 1450
C TO A AND B                                                           LZI 1460
        IF (I.EQ.M) GO TO 170                                          LZI 1470
        W = A(I,I-1)                                                   LZI 1480
        Z = A(J,I-1)                                                   LZI 1490
        D1 = ABS(REAL(Z)) + ABS(AIMAG(Z))                              LZI 1500
        D2 = ABS(REAL(W)) + ABS(AIMAG(W))                              LZI 1510
        IF (D1.EQ.0.0) GO TO 60                                        LZI 1520
  170   IF (D2.GT.D1) GO TO 190                                        LZI 1530
C MUST INTERCHANGE ROWS                                                LZI 1540
        DO 180 K=I,NNORN                                               LZI 1550
          Y = A(I,K)                                                   LZI 1560
          A(I,K) = A(J,K)                                              LZI 1570
          A(J,K) = Y                                                   LZI 1580
          Y = B(I,K)                                                   LZI 1590
```

```
            B(I,K) = B(J,K)                                                 LZI 1600
            B(J,K) = Y                                                      LZI 1610
  180     CONTINUE                                                          LZI 1620
          IF (I.GT.M) A(I,I-1) = A(J,I-1)                                   LZI 1630
          IF (D2.EQ.0.0) GO TO 220                                          LZI 1640
C THE SCALING OF W AND Z IS DESIGNED TO AVOID A DIVISION BY ZERO            LZI 1650
C WHEN THE DENOMINATOR IS SMALL                                             LZI 1660
          Y = CMPLX(REAL(W)/D1,AIMAG(W)/D1)/CMPLX(REAL(Z)/D1,AIMAG(Z)/      LZI 1670
         *  D1)                                                             LZI 1680
          GO TO 200                                                         LZI 1690
  190     Y = CMPLX(REAL(Z)/D2,AIMAG(Z)/D2)/CMPLX(REAL(W)/D2,AIMAG(W)/      LZI 1700
         *  D2)                                                             LZI 1710
  200     DO 210 K=I,NNORN                                                  LZI 1720
            A(J,K) = A(J,K) - Y*A(I,K)                                      LZI 1730
            B(J,K) = B(J,K) - Y*B(I,K)                                      LZI 1740
  210     CONTINUE                                                          LZI 1750
          IF (I.GT.M) A(J,I-1) = (0.0,0.0)                                  LZI 1760
C PERFORM TRANSFORMATIONS FROM RIGHT TO RESTORE B TO                        LZI 1770
C   TRIANGLULAR FORM                                                        LZI 1780
C APPLY TRANSFORMATIONS TO A                                                LZI 1790
  220     Z = B(J,I)                                                        LZI 1800
          W = B(J,J)                                                        LZI 1810
          D2 = ABS(REAL(W)) + ABS(AIMAG(W))                                 LZI 1820
          D1 = ABS(REAL(Z)) + ABS(AIMAG(Z))                                 LZI 1830
          IF (D1.EQ.0.0) GO TO 60                                           LZI 1840
          IF (D2.GT.D1) GO TO 270                                           LZI 1850
C MUST INTERCHANGE COLUMNS                                                  LZI 1860
          DO 230 K=LOR1,J                                                   LZI 1870
            Y = A(K,J)                                                      LZI 1880
            A(K,J) = A(K,I)                                                 LZI 1890
            A(K,I) = Y                                                      LZI 1900
            Y = B(K,J)                                                      LZI 1910
            B(K,J) = B(K,I)                                                 LZI 1920
            B(K,I) = Y                                                      LZI 1930
  230     CONTINUE                                                          LZI 1940
          IF (I.EQ.NM1) GO TO 240                                           LZI 1950
          Y = A(J+1,J)                                                      LZI 1960
          A(J+1,J) = A(J+1,I)                                               LZI 1970
          A(J+1,I) = Y                                                      LZI 1980
  240     IF (.NOT.WANTX) GO TO 260                                         LZI 1990
          DO 250 K=1,N                                                      LZI 2000
            Y = X(K,J)                                                      LZI 2010
            X(K,J) = X(K,I)                                                 LZI 2020
            X(K,I) = Y                                                      LZI 2030
  250     CONTINUE                                                          LZI 2040
  260     IF (D2.EQ.0.0) GO TO 310                                          LZI 2050
          Z = CMPLX(REAL(W)/D1,AIMAG(W)/D1)/CMPLX(REAL(Z)/D1,AIMAG(Z)/      LZI 2060
         *  D1)                                                             LZI 2070
          GO TO 280                                                         LZI 2080
  270     Z = CMPLX(REAL(Z)/D2,AIMAG(Z)/D2)/CMPLX(REAL(W)/D2,AIMAG(W)/      LZI 2090
         *  D2)                                                             LZI 2100
  280     DO 290 K=LOR1,J                                                   LZI 2110
            A(K,I) = A(K,I) - Z*A(K,J)                                      LZI 2120
            B(K,I) = B(K,I) - Z*B(K,J)                                      LZI 2130
  290     CONTINUE                                                          LZI 2140
          B(J,I) = (0.0,0.0)                                                LZI 2150
          IF (I.LT.NM1) A(I+2,I) = A(I+2,I) - Z*A(I+2,J)                    LZI 2160
          IF (.NOT.WANTX) GO TO 310                                         LZI 2170
          DO 300 K=1,N                                                      LZI 2180
            X(K,I) = X(K,I) - Z*X(K,J)                                      LZI 2190
  300     CONTINUE                                                          LZI 2200
  310   CONTINUE                                                            LZI 2210
        GO TO 60                                                            LZI 2220
  320   CONTINUE                                                            LZI 2230
        EIGA(NN) = A(NN,NN)                                                 LZI 2240
        EIGB(NN) = B(NN,NN)                                                 LZI 2250
        IF (NN.EQ.1) GO TO 330                                              LZI 2260
        ITER(NN) = ITS                                                      LZI 2270
        NN = NM1                                                            LZI 2280
        IF (NN.GT.1) GO TO 50                                               LZI 2290
        ITER(1) = 0                                                         LZI 2300
        GO TO 320                                                           LZI 2310
C FIND EIGENVECTORS USING B FOR INTERMEDIATE STORAGE                        LZI 2320
  330   IF (.NOT.WANTX) RETURN                                              LZI 2330
        M = N                                                               LZI 2340
  340   CONTINUE                                                            LZI 2350
        ALFM = A(M,M)                                                       LZI 2360
        BETM = B(M,M)                                                       LZI 2370
        B(M,M) = (1.0,0.0)                                                  LZI 2380
        L = M - 1                                                           LZI 2390
        IF (L.EQ.0) GO TO 370                                               LZI 2400
  350   CONTINUE                                                            LZI 2410
        L1 = L + 1                                                          LZI 2420
        SL = (0.0,0.0)                                                      LZI 2430
        DO 360 J=L1,M                                                       LZI 2440
          SL = SL + (BETM*A(L,J)-ALFM*B(L,J))*B(J,M)                        LZI 2450
  360   CONTINUE                                                            LZI 2460
        Y = BETM*A(L,L) - ALFM*B(L,L)                                       LZI 2470
        IF (REAL(Y).EQ.0.0 .AND. AIMAG(Y).EQ.0.0) Y =                       LZI 2480
       * CMPLX((EPSA+EPSB)/2.0,0.0)                                        LZI 2490
        B(L,M) = -SL/Y                                                      LZI 2500
        L = L - 1                                                           LZI 2510
```

```
  370 IF (L.GT.0) GO TO 350                                          LZI 2520
      M = M - 1                                                      LZI 2530
      IF (M.GT.0) GO TO 340                                          LZI 2540
C TRANSFORM TO ORIGINAL COORDINATE SYSTEM                            LZI 2550
      M = N                                                          LZI 2560
  380 CONTINUE                                                       LZI 2570
      DO 400 I=1,N                                                   LZI 2580
        S = (0.0,0.0)                                                LZI 2590
        DO 390 J=1,M                                                 LZI 2600
          S = S + X(I,J)*B(J,M)                                      LZI 2610
  390   CONTINUE                                                     LZI 2620
        X(I,M) = S                                                   LZI 2630
  400 CONTINUE                                                       LZI 2640
      M = M - 1                                                      LZI 2650
      IF (M.GT.0) GO TO 380                                          LZI 2660
C NORMALIZE SO THAT LARGEST COMPONENT = 1.                           LZI 2670
      M = N                                                          LZI 2680
  410 CONTINUE                                                       LZI 2690
      SS = 0.                                                        LZI 2700
      DO 420 I=1,N                                                   LZI 2710
        R = ABS(REAL(X(I,M))) + ABS(AIMAG(X(I,M)))                   LZI 2720
        IF (R.LT.SS) GO TO 420                                       LZI 2730
        SS = R                                                       LZI 2740
        D = X(I,M)                                                   LZI 2750
  420 CONTINUE                                                       LZI 2760
      IF (SS.EQ.0.0) GO TO 440                                       LZI 2770
      DO 430 I=1,N                                                   LZI 2780
        X(I,M) = X(I,M)/D                                            LZI 2790
  430 CONTINUE                                                       LZI 2800
  440 M = M - 1                                                      LZI 2810
      IF (M.GT.0) GO TO 410                                          LZI 2820
      RETURN                                                         LZI 2830
      END                                                            LZI 2840
```

ACM Transactions on Mathematical Software, Vol. 2, No. 4, December 1976, Page 396.

REMARK ON ALGORITHM 496

The LZ Algorithm to Solve the Generalized Eigenvalue Problem for Complex Matrices [F2]

[L.C. Kaufman, *ACM Trans. Math. Software 1*, 3 (Sept. 1975), 271–281]

L.C. Kaufman [Recd 22 April 1976]
Bell Laboratories, Murray Hill, NJ 07974.

The following incorrect lines should be changed:

from

```
      IF(I.GT.M) A(J,I-1)=(0.0,0.0)          LZI 1760
220   Z=B(J,I)                               LZI 1800
```

to

```
220   IF(I.GT.M) A(J,I--1)=(0.0,0.0)         LZI 1760
      Z=B(J,I)                               LZI 1800
```

The following unnecessary line should be deleted:

```
      NP1=N+1                                LZI 1360
```

ALGORITHM 497
Automatic Integration of Functional Differential Equations [D2]

KENNETH W. NEVES

Babcock & Wilcox

Key Words and Phrases: functional differential equations, automatic integration, one-step and multistep methods, time lags and retarded arguments, Volterra integro-differential equations, derivative jump discontinuities
CR Categories: 5.16, 5.17, 5.18
Language: Fortran

DESCRIPTION

This algorithm is a complement to [1] where the theoretical background and development are described.

REFERENCES

1. NEVES, K.W. Automatic integration of functional differential equations: an approach. *ACM Trans. Math. Software 1*, 4 (Dec. 1975), 357–368.

ALGORITHM

```
      SUBROUTINE DMRODE(F, A, B, HMIN, H, HMAX, N, X0, WK, EPS,          DMR   10
     * JSTART, AA, TT, XX, ZZ, FINIT, ALPHA, WKA, IER, IQ)               DMR   20
C                                                                        DMR   30
C F     -AUTOMATIC STEP CHANGE MERSON DIFFERENTIAL EQUATION              DMR   40
C U     -  SOLVER MODIFIED TO HANDLE FUNCTIONAL DIFFERENTIAL             DMR   50
C N     -  EQUATIONS OF THE FORM                                         DMR   60
C C     -        (1) DX/DT =F(T,X,Z,I)                                   DMR   70
C T     -  WHERE F (DESCRIBED BELOW) IS A VECTOR VALUED                  DMR   80
C I     -  FUNCTION OF DIMENSION N, X AND Z ARE                          DMR   90
C O     -  N VECTORS SUCH THAT THE ITH COMPONENT OF Z IS                 DMR  100
C N     -  THE ITH COMPONENT OF X EVALUATED AT ALPHA(X,T,I)              DMR  110
C       -  ALPHA IS A VECTOR VALUED FUNCTION (DESCRIBED                  DMR  120
C       -  BELOW). ALPHA IS A USER SUPPLIED STATE DEPENDENT              DMR  130
C       -  RETARDING FUNCTION, AND (1) IS CALLED A                       DMR  140
C       -  RETARDED ORDINARY DIFF. EQ. (RODE) WITH STATE                 DMR  150
C       -  DEPENDENT LAG (SDL). NOTE -AS WRITTEN ABOVE                   DMR  160
C       -  EACH COMPONENT OF DX/DT MAY DEPEND ON A DIFFERENT             DMR  170
C       -  RETARDING FUNCTION.  IT IS ALSO POSSIBLE TO ALLOW             DMR  180
C       -  A GIVEN COMPONENT TO DEPEND ON MORE THAN ONE                  DMR  190
C       -  RETARDING FUNCTION BY AUGMENTING THE ORDER OF                 DMR  200
C       -  THE SYSTEM (SEE  DISCRIPTION  FOR DETAILS).                   DMR  210
C P    F-USER SUPPLIED EXTERNAL FUNCTION. THE DERIVATIVE                 DMR  220
C A     -  OF THE ITH COMPONENT OF X IS F(T, X, Z,I), WHERE              DMR  230
C R     -  Z  AND X  ARE ARRAYS OF DIMENSION N.   X(I)                   DMR  240
```

Received 8 May 1975.

A version of this paper was presented at Mathematical Software II, a conference held at Purdue University, West Lafayette, Indiana, May 29–31, 1974.
Author's address: Babcock & Wilcox Co., P.O. Box 1260, Lynchburg, VA. 24505.

```
C A     -  IS THE SOLUTION OF THE ITH COMPONENET AT T AND                DMR  250
C M     -  Z (I) IS THE ITH COMPONENT OF X AT ALPHA(X,T,I)               DMR  260
C E     -  Z(I) IS CALCULATED INTERNALLY WITHIN DMRODE AND               DMR  270
C T     -  PASSED TO F (SEE ALPHA BELOW). T IS THE CURRENT               DMR  280
C E     -  VALUE OF THE INDEPENDENT VARIABLE.                            DMR  290
C R    A-STARTING POINT OF INTEGRATION                                   DMR  300
C S    B-FINAL POINT OF INTEGRATION  (NOT NEC. .GT. A)                   DMR  310
C   HMIN-THE ABSOLUTE VALUE OF MINIMUM ALLOWABLE STEP SIZE               DMR  320
C       -  THE STEP SIZE WILL NEVER BE SMALLER THAN HMIN                 DMR  330
C       -  IN ABSOLUTE VALUE EXCEPT AT THE END OF THE                    DMR  340
C       -INTERVAL WHERE H IS ALWAYS .GT. HMIN/2 IN ABS..                 DMR  350
C      H-INITIAL GUESS OF THE INTEGRATION STEP SIZE. THE                 DMR  360
C       -  ADJUSTED STEPSIZE IS RETURNED IN H.                           DMR  370
C   HMAX-THE ABSOLUTE VALUE OF MAXIMUM ALLOWABLE STEP SIZE               DMR  380
C      N-NUMBER OF SIMULTANEOUS EQUATIONS                                DMR  390
C     X0-VECTOR OF LENGTH N CONTAINING THE VALUE OF X(A).                DMR  400
C       -  X(B) IS RETURNED IN X0.                                       DMR  410
C     WK-WORK AREA OF DIMENSION .GE.      4*N                            DMR  420
C    EPS-INPUT REQUEST OF RELATIVE ERROR IN LARGEST COMP.                DMR  430
C JSTART-A PARAMETER USED TO INTIALIZE Z ARRAY INTERNALLY                 DMR  440
C       -  IN DMRODE. JSTART MUST BE SET EQUAL TO ZERO ON                DMR  450
C       -  FIRST CALL TO DMRODE. TO INTEGRATE BEYOND B AFTER             DMR  460
C       -  FIRST CALL CERTAIN ARRAYS MUST BE SAVED. THIS                 DMR  470
C       -  IS DONE AUTOMATICALLY BY LETTING JSTART=1.                    DMR  480
C       - IF JSTART=-1, THEN THE INITIAL CONDITIONS AT                   DMR  490
C       -  THE PREVIOUS CALL ARE REINSTATED AND THE INTE-                DMR  500
C       -  GRATION FROM A TO POSSIBLY A NEW CHOICE FOR B                 DMR  510
C       -  CAN BE EXECUTED.  THIS ALLOWS INTERATIVE USE OF               DMR  520
C       -  DMRODE IN ORDER TO CALCULATE DERIVATIVE JUMP                  DMR  530
C       -  POINTS IN THE SOLUTION(SEE DESCRIPTION)                       DMR  540
C     AA-A PARAMETER THAT MUST BE SET EQUAL TO THE VALUE OF              DMR  550
C       -  THE STARTING POINT A WHEN FIRST CALL TO DMRODE                DMR  560
C       -  WAS MADE (I.E. WHEN JSTART=0) (SEE FINIT)                     DMR  570
C     TT-STORAGE ARRAY FOR SAVING TIME STEP VALUES                       DMR  580
C       -  TT IS OF DIMENSION IQ WHERE IQ IS .GT.LENGTH OF               DMR  590
C       -  ENTIRE INTERVAL/HMIN                                          DMR  600
C  XX,ZZ-STORAGE ARRAYS OF DIMENSION N BY IQ USED FOR                    DMR  610
C       -  SAVING VALUES OF X AND Z DESCRIBED IN F.                      DMR  620
C  FINIT-USER SUPPLIED EXTERNAL FUNCTION, FINIT(T,I). FINIT              DMR  630
C       -  RETURNS THE VALUE OF THE ITH COMPONENT OF                     DMR  640
C       -  THE INITIAL FUNCTION AT T WHENEVER ((T. LE.AA)                DMR  650
C       -  .AND.(H.GE.0)).OR.((T.GE.AA).AND.(H.LE.0)).                   DMR  660
C  ALPHA-USER SUPPLIED EXTERNAL FUNCTION OF THE                          DMR  670
C       -  FORM ALPHA(X,T,I) WHERE X IS ASSUMED TO BE                    DMR  680
C       -  THE CURRENT SOLUTION (SUPPLIED BY DMRODE) AND                 DMR  690
C       -  T THE CURRENT TIME.  IN GENERAL ALPHA WILL BE                 DMR  700
C       -  DIFFERENT FOR EACH COMPONENT X(I).  IN THE EVENT              DMR  710
C       -  A RODE IS POSED WITH MORE THAN ONE LAG IN ANY                 DMR  720
C       -  GIVEN COMPONENT THE SYSTEM MAY BE AUGMENTED TO                DMR  730
C       -  HANDLE THIS SITUATION (SEE DISCRIPTION).                      DMR  740
C    WKA-WORK AREA OF DIMENSION .GE.11*N UNLESS                          DMR  750
C       -  IT IS KNOWN THE LAG T-ALPHA(T,X(T),I) IN ALL                  DMR  760
C       -  COMPONENTS WILL ALWAYS EXCEED HMAX OR VANISH                  DMR  770
C       -  OVER THE ENTIRE INTERVAL IN A GIVEN COMPONENT,                DMR  780
C       -  IN WHICH CASE A DIMENSION OF 3*N WILL SUFFICE                 DMR  790
C    IER-OUTPUT PARAMETER IF IER=0 AND HMAX.NE.HMIN                      DMR  800
C       -  NORMAL ERROR CHECKING TOOK PLACE. IF IER=1                    DMR  810
C       -  HMIN WAS ATTAINED BY THE STEP CHANGING                        DMR  820
C       -  PROCEDURE, HENCE REQUESTED ACCURACY IS NOT                    DMR  830
C       -  GUARANTEED                                                    DMR  840
C     IQ-THE DIMENSION OF TT IN THE CALLING PROGRAM (ALSO                DMR  850
C       -  THE DIMENSION OF THE 2ND ARGUMENT OF THE XX AND               DMR  860
C       -  ZZ ARRAYS).                                                   DMR  870
C  PRECISION - SINGLE                                                    DMR  880
C  AUTHOR/IMPLEMENTOR- KENNETH W. NEVES                                  DMR  890
C  REQUIRED ROUTINES                                                     DMR  900
C       - DZETA, AN INTERPOLATION SCHEME OF ORDER COMP-                  DMR  910
C       -  ARABLE TO 4TH ORDER MERSON METHOD. DZETA IN TURN              DMR  920
C       -  WILL CALL STARTER METHOD WHEN LAG IS SMALLER                  DMR  930
C       -  THAN CURRENT STEP SIZE.                                       DMR  940
C                                                                        DMR  950
      DIMENSION WK(1), X0(1)                                             DMR  960
      DIMENSION WKA(1)                                                   DMR  970
      DIMENSION TT(1), XX(N,1), ZZ(N,1)                                  DMR  980
      EXTERNAL F, FINIT, ALPHA                                           DMR  990
      INTEGER SW                                                         DMR 1000
      LOGICAL BE, BH, BR, BX, BT                                         DMR 1010
      DATA ZERO, P5, OP5, THREE, FOUR /0.,.5,1.5,3.,4./                  DMR 1020
      E5 = .5*EPS                                                        DMR 1030
      IER = 0                                                            DMR 1040
      IF (A.EQ.B) GO TO 290                                              DMR 1050
      IB1 = N + N                                                        DMR 1060
      IB2 = IB1 + N                                                      DMR 1070
C                     ************************************               DMR 1080
C                      CHECK FOR THE PROPER SIGN OF H                    DMR 1090
C                     ************************************               DMR 1100
      H = SIGN(ABS(H),B-A)                                               DMR 1110
C                     ************************************               DMR 1120
C                      CHECK FOR 1ST CALL TO DMRODE. IF                  DMR 1130
C                      YES, THEN INITIALIZE NEC. ARRAYS                  DMR 1140
C                     ************************************               DMR 1150
```

```
      IF (JSTART.NE.0) GO TO 20                                          DMR 1160
      JRAY = 1                                                           DMR 1170
      JSAVE = JRAY                                                       DMR 1180
      TT(1) = A                                                          DMR 1190
      DO 10 J=1,N                                                        DMR 1200
        XX(J,1) = X0(J)                                                  DMR 1210
          ZZ(J,1)=DZETA(TT(1),X0,TT,XX,ZZ,AA,F,FINIT,ALPHA,              DMR 1220
     *    N,J,WKA(N+1),H,1,IQ)                                           DMR 1230
   10 CONTINUE                                                           DMR 1240
   20 CONTINUE                                                           DMR 1250
      BH = .TRUE.                                                        DMR 1260
      BR = .TRUE.                                                        DMR 1270
      BX = .TRUE.                                                        DMR 1280
      BT = .TRUE.                                                        DMR 1290
      X = A                                                              DMR 1300
      XS = A                                                             DMR 1310
      IF (ABS(H).GE.HMAX) H = SIGN(1.,H)*HMAX                            DMR 1320
      DO 30 J=1,N                                                        DMR 1330
        IJK0 = N + J                                                     DMR 1340
        WK(IJK0) = X0(J)                                                 DMR 1350
   30 CONTINUE                                                           DMR 1360
C                    ************************************                DMR 1370
C                     CHECK JSTART AND EXECUTE                           DMR 1380
C                     PROPER INITIALIZATION                              DMR 1390
C                    ************************************                DMR 1400
      IF (JSTART) 40, 90, 50                                             DMR 1410
   40 JRAY = JSAVE                                                       DMR 1420
   50 DO 60 J=1,N                                                        DMR 1430
        IJK0 = N + J                                                     DMR 1440
        WK(IJK0) = XX(J,JRAY)                                            DMR 1450
   60 CONTINUE                                                           DMR 1460
      JSAVE = JRAY                                                       DMR 1470
      GO TO 90                                                           DMR 1480
   70 XS = X                                                             DMR 1490
C                    ************************************                DMR 1500
C                     AFTER EACH SUCCESSFUL TIME STEP                    DMR 1510
C                     UPDATE TT,XX,ZZ  ARRAYS. CALL TO                   DMR 1520
C                     DZETA PERFORMS NECESSARY INTERPO-                  DMR 1530
C                     LATION AT LAG POINT.                               DMR 1540
C                    ************************************                DMR 1550
      TT(JRAY+1) = X                                                     DMR 1560
      DO 80 J=1,N                                                        DMR 1570
        IJK0 = N + J                                                     DMR 1580
        WK(IJK0) = X0(J)                                                 DMR 1590
        XX(J,JRAY+1) = X0(J)                                             DMR 1600
        HH = X - TT(JRAY)                                                DMR 1610
        ZZ(J,JRAY+1) = DZETA(TT(JRAY+1),X0,TT,XX,ZZ,AA,F,FINIT,          DMR 1620
     *   ALPHA,N,J,WKA(N+1),HH,JRAY,IQ)                                  DMR 1630
   80 CONTINUE                                                           DMR 1640
C                    ************************************                DMR 1650
C                     INSERT CALL TO PRINT ROUTINE HERE                  DMR 1660
C                     IF DESIRED.                                        DMR 1670
C                    ************************************                DMR 1680
      JRAY = JRAY + 1                                                    DMR 1690
      IF (BR) GO TO 90                                                   DMR 1700
      H = HSS                                                            DMR 1710
      RETURN                                                             DMR 1720
   90 HSS = H                                                            DMR 1730
C                    ************************************                DMR 1740
C                     TEST FOR END OF PROBLEM                            DMR 1750
C                    ************************************                DMR 1760
      Q = X + H - B                                                      DMR 1770
      BE = .TRUE.                                                        DMR 1780
      IF (.NOT.((H.GT.ZERO .AND. Q/H.GE.-.5) .OR. (H.LT.ZERO .AND.       DMR 1790
     * Q/(-H).LE..5))) GO TO 110                                         DMR 1800
      IF (BT) HSS = H                                                    DMR 1810
      H = .5*(B-X)                                                       DMR 1820
      IF (.NOT.BT) H = B - X                                             DMR 1830
      IF (BT) GO TO 100                                                  DMR 1840
      BR = .FALSE.                                                       DMR 1850
  100 BT = .FALSE.                                                       DMR 1860
  110 CONTINUE                                                           DMR 1870
      H3 = H/THREE                                                       DMR 1880
C                    ************************************                DMR 1890
C                     CALCULATE SOLN. AT  X+H                            DMR 1900
C                    ************************************                DMR 1910
      DO 270 SW=1,5                                                      DMR 1920
C                    ************************************                DMR 1930
C                     DZETA CALCULATES LAG AND SOLUTION                  DMR 1940
C                     AT    PAST     POINT (NEC. FOR F )                 DMR 1950
C                    ************************************                DMR 1960
        DO 120 JS=1,N                                                    DMR 1970
          WKA(JS) = DZETA(X,X0,TT,XX,ZZ,AA,F,FINIT,ALPHA,N,JS,           DMR 1980
     *     WKA(N+1),H,JRAY,IQ)                                           DMR 1990
  120   CONTINUE                                                         DMR 2000
        DO 130 JS=1,N                                                    DMR 2010
          WK(JS) = F(X,X0,WKA,JS)                                        DMR 2020
  130   CONTINUE                                                         DMR 2030
        DO 230 I=1,N                                                     DMR 2040
          Q = H3*WK(I)                                                   DMR 2050
          IJK0 = N + I                                                   DMR 2060
```

```
            IJK1 = IB1 + I                                                  DMR 2070
            IJK2 = IB2 + I                                                  DMR 2080
            GO TO (140, 150, 160, 170, 180), SW                             DMR 2090
  140       R = Q                                                           DMR 2100
            WK(IJK1) = Q                                                    DMR 2110
            GO TO 190                                                       DMR 2120
C                                                                           DMR 2130
  150       R = P5*(Q+WK(IJK1))                                             DMR 2140
            GO TO 190                                                       DMR 2150
C                                                                           DMR 2160
  160       R = THREE*Q                                                     DMR 2170
            WK(IJK2) = R                                                    DMR 2180
            R = .375*(R+WK(IJK1))                                           DMR 2190
            GO TO 190                                                       DMR 2200
C                                                                           DMR 2210
  170       R = WK(IJK1) + FOUR*Q                                           DMR 2220
            WK(IJK1) = R                                                    DMR 2230
            R = OP5*(R-WK(IJK2))                                            DMR 2240
            GO TO 190                                                       DMR 2250
C                                                                           DMR 2260
  180       R = P5*(Q+WK(IJK1))                                             DMR 2270
            Q = ABS(R+R-OP5*(Q+WK(IJK2)))                                   DMR 2280
  190       X0(I) = WK(IJK0) + R                                            DMR 2290
            IF (SW.NE.5) GO TO 230                                          DMR 2300
C                    ************************************                   DMR 2310
C                     AUTOMATIC STEP CHANGE                                 DMR 2320
C                    ************************************                   DMR 2330
            E = ABS(X0(I))                                                  DMR 2340
            R = E5                                                          DMR 2350
            IF (E.GE.1.E-3) R = E*E5                                        DMR 2360
            IF (HMIN.EQ.HMAX) GO TO 280                                     DMR 2370
C                    ************************************                   DMR 2380
C                     TEST ADJUSTMENT OF THE STEP                           DMR 2390
C                    ************************************                   DMR 2400
            IF (Q.LT.R .OR. (.NOT.BX)) GO TO 220                            DMR 2410
            BR = .TRUE.                                                     DMR 2420
            BT = .TRUE.                                                     DMR 2430
            BH = .FALSE.                                                    DMR 2440
            H = P5*H                                                        DMR 2450
            IF (ABS(H).GT.HMIN) GO TO 200                                   DMR 2460
            IF (.NOT.BR) GO TO 280                                          DMR 2470
C                    ************************************                   DMR 2480
C                     THE STEP IS HALVED RESTORE X AND X0,                  DMR 2490
C                     AND GO BACK FOR REPEATED INTEGRATION                  DMR 2500
C                     WITH THIS NEW STEP                                    DMR 2510
C                    ************************************                   DMR 2520
            H = SIGN(1.,H)*HMIN                                             DMR 2530
            BX = .FALSE.                                                    DMR 2540
            IER = 1                                                         DMR 2550
  200       DO 210 J=1,N                                                    DMR 2560
              IJK0 = N + J                                                  DMR 2570
              X0(J) = WK(IJK0)                                              DMR 2580
  210       CONTINUE                                                        DMR 2590
            X = XS                                                          DMR 2600
            GO TO 90                                                        DMR 2610
C                                                                           DMR 2620
  220       IF (Q.GE.0.03125*R) BE = .FALSE.                                DMR 2630
  230     CONTINUE                                                          DMR 2640
          GO TO (240, 270, 250, 260, 270), SW                               DMR 2650
  240     X = X + H3                                                        DMR 2660
          GO TO 270                                                         DMR 2670
C                                                                           DMR 2680
  250     X = X + P5*H3                                                     DMR 2690
          GO TO 270                                                         DMR 2700
C                                                                           DMR 2710
  260     X = X + P5*H                                                      DMR 2720
  270   CONTINUE                                                          DMR 2730
C                    ************************************                   DMR 2740
C                     TEST A POSSIBLE DOUBLING OF THE STEP                  DMR 2750
C                    ************************************                   DMR 2760
        IF (.NOT.(BE .AND. BH .AND. BR)) GO TO 280                          DMR 2770
        H = H + H                                                           DMR 2780
        BX = .TRUE.                                                         DMR 2790
  280   BH = .TRUE.                                                       DMR 2800
        GO TO 70                                                          DMR 2810
C                                                                           DMR 2820
  290   DO 300 I=1,N                                                      DMR 2830
          X0(I) = ZERO                                                    DMR 2840
  300   CONTINUE                                                          DMR 2850
        RETURN                                                            DMR 2860
        END                                                               DMR 2870


      FUNCTION DZETA(TA, XA, T, X, XL, TAU, F, FINIT, ALPHA, N,           DZE   10
     * JCOMP, W, HP, JRAY, NN)                                            DZE   20
C                                                                         DZE   30
C F     -INTERPOLATION ROUTINE. CALLED BY DMRODE AND                      DZE   40
C U     -  REQUIRES NO USER INTERFACE OTHER THAN USER                     DZE   50
C N     -  SUPPLIED EXTERNAL FUNCTION ALPHA (SEE DMRODE).                 DZE   60
C C     -  DZETA CALCULATES THE LAG(OR RETARDATION) VIA                   DZE   70
```

```
C T     -  ALPHA WITH TA,XA AS INPUT, THEN USES 2-POINT                DZE   80
C I     -  HERMITE INTERPOLATION TO APPROXIMATE SOLUTION               DZE   90
C O     -  AT LAG. IF STEP SIZE IS LARGER THAN LAG,HERMITE             DZE  100
C N     -  INTERPOLANT CANNOT BE OBTAINED, SO DZETA CALLS              DZE  110
C       -  A TRUE ONESTEP FDE SOLVER. SINCE DMRODE CALLS               DZE  120
C       -  DZETA THE PARAMETER DESCRIPTION CAN BE FOUND                DZE  130
C       -  IN DMRODE                                                   DZE  140
      DIMENSION W(1), XA(1)                                            DZE  150
      DIMENSION T(1), X(N,1), XL(N,1)                                  DZE  160
      EXTERNAL F, FINIT, ALPHA                                         DZE  170
      DATA J /1/                                                       DZE  180
      IF ((J.GE.JRAY) .AND. (J.NE.1)) J = JRAY - 1                     DZE  190
      IF (J.LE.0) J = 1                                                DZE  200
C                     ************************************             DZE  210
C                      CALCULATE LAG                                   DZE  220
C                     ************************************             DZE  230
      B = ALPHA(XA,TA,JCOMP)                                           DZE  240
      IF (B.EQ.TA) DZETA = XA(JCOMP)                                   DZE  250
      IF (B.EQ.TA) RETURN                                              DZE  260
      IF (((HP.LT.0.) .AND. (B.GE.TAU)) .OR. ((HP.GE.0.) .AND.         DZE  270
     * (B.LE.TAU))) GO TO 100                                          DZE  280
C                     ************************************             DZE  290
C                      SEARCH T ARRAY IN ORDER TO FIND                 DZE  300
C                      INTERVAL CONTAINING ALPHA(XA,TA,I)..            DZE  310
C                      IF NO SUCH INTERVAL IS FOUND CALL               DZE  320
C                      START   OR IF ARRAY EXCEEDS                     DZE  330
C                      DIMENSION IQ OF DMRODE, STOP                    DZE  340
C                     ************************************             DZE  350
      DX = B - T(J)                                                    DZE  360
      IF (HP.LT.0.) DX = -DX                                           DZE  370
      IF (DX) 10, 40, 30                                               DZE  380
   10 IF (J.EQ.1) GO TO 40                                             DZE  390
      J = J - 1                                                        DZE  400
      DX = B - T(J)                                                    DZE  410
      IF (HP.LT.0.) DX = -DX                                           DZE  420
      IF (DX) 10, 40, 40                                               DZE  430
   20 J = J + 1                                                        DZE  440
   30 IF (J.EQ.NN) GO TO 50                                            DZE  450
      IF (J+1.GT.JRAY) GO TO 90                                        DZE  460
      DDX = B - T(J+1)                                                 DZE  470
      IF (HP.LT.0.) DDX = -DDX                                         DZE  480
      IF (DDX) 40, 20, 20                                              DZE  490
   40 IF (J.EQ.NN) GO TO 50                                            DZE  500
      GO TO 60                                                         DZE  510
C                                                                      DZE  520
   50 STOP                                                             DZE  530
   60 CONTINUE                                                         DZE  540
C                     ************************************             DZE  550
C                     2-POINT HERMITE INTERPOLATION                    DZE  560
C                     ************************************             DZE  570
      IF (T(J+1).GE.TA) GO TO 90                                       DZE  580
      DX = T(J+1) - T(J)                                               DZE  590
      HH = B - T(J)                                                    DZE  600
      DO 70 I=1,N                                                      DZE  610
        II = I + N                                                     DZE  620
        W(I) = X(I,J)                                                  DZE  630
        W(II) = XL(I,J)                                                DZE  640
   70 CONTINUE                                                         DZE  650
      AC = F(T(J),W,W(N+1),JCOMP)                                      DZE  660
      DO 80 I=1,N                                                      DZE  670
        II = I + N                                                     DZE  680
        W(I) = X(I,J+1)                                                DZE  690
        W(II) = XL(I,J+1)                                              DZE  700
   80 CONTINUE                                                         DZE  710
      AF2P = F(T(J+1),W,W(N+1),JCOMP)                                  DZE  720
      DIVDF1 = (X(JCOMP,J+1)-X(JCOMP,J))/DX                            DZE  730
      DIVDF3 = AC + AF2P - 2.*DIVDF1                                   DZE  740
      C3 = (DIVDF1-AC-DIVDF3)/DX                                       DZE  750
      C4 = DIVDF3/DX**2                                                DZE  760
      DZETA = X(JCOMP,J) + HH*(AC+HH*(C3+HH*C4))                       DZE  770
      GO TO 110                                                        DZE  780
C                                                                      DZE  790
   90 CONTINUE                                                         DZE  800
      J = JRAY                                                         DZE  810
      CALL START(TA, XA, T, X, XL, F, FINIT, ALPHA, N, JCOMP, W, J,    DZE  820
     * ANS, HP)                                                        DZE  830
      DZETA = ANS                                                      DZE  840
      RETURN                                                           DZE  850
C                                                                      DZE  860
  100 DZETA = FINIT(B,JCOMP)                                           DZE  870
  110 CONTINUE                                                         DZE  880
      RETURN                                                           DZE  890
C                                                                      DZE  900
C                                                                      DZE  910
      END                                                              DZE  920


      SUBROUTINE START(TA, XA, T, X, XL, F, FINT, ALPHA, N, JCOMP,     STA   10
     * W, JEND, ANS, H1)                                              STA   20
C F     -START IS A SUBROUTINE THAT CAN APPROXIMATE THE                STA   30
```

```
C U     -  SOLUTION AT A PAST POINT GIVEN ONLY THE SOLUTION          STA   40
C N     -  AT THE MOST RECENT MESH POINT AND THE FUNCTIONS           STA   50
C C     -  F AND ALPHA, AS DESCRIBED BY DMRODE.  IT IS A             STA   60
C T     -  RUNGE-KUTTA TYPE ALGORITHM MODIFIED FOR RODES             STA   70
C I     -  AND IS GLOBALLY 3RD ORDER, BUT PREDICTS 4TH               STA   80
C O     -  ORDER APPROX. TO BE USED IN EVALUATION OF F               STA   90
C N     -  IN MERSON METHOD OF DMRODE.  START, DZETA, AND            STA  100
C       -  DMRODE COMBINE TO GIVE 4TH ORDER SIMPLIFIED               STA  110
C       -  PREDICTOR-CORRECTOR TWO STEP ALGORITHM, WHERE             STA  120
C       -  START IS A TRUE ONE-STEP PREDICTOR USED ONLY              STA  130
C       -  WHEN STEP SIZE EXCEEDS THE LAG                            STA  140
      DIMENSION T(1), X(N,1), XL(N,1), XA(1), W(1)                   STA  150
      I1 = N                                                         STA  160
      I2 = I1 + N                                                    STA  170
      I3 = I2 + N                                                    STA  180
      I4 = I3 + N                                                    STA  190
      I5 = I4 + N                                                    STA  200
      I6 = I5 + N                                                    STA  210
      I7 = I6 + N                                                    STA  220
      I8 = I7 + N                                                    STA  230
      I9 = I8 + N                                                    STA  240
      JRAY = JEND                                                    STA  250
C                    ************************************           STA  260
C                     RECALCULATE LAG                               STA  270
C                    ************************************           STA  280
      B = ALPHA(XA,TA,JCOMP)                                         STA  290
   10 TEST = B - T(JRAY)                                             STA  300
      IF (H1.LT.0.) TEST = -TEST                                     STA  310
      IF (TEST.GT.0.) GO TO 20                                       STA  320
      JRAY = JRAY - 1                                                STA  330
      GO TO 10                                                       STA  340
C                    ************************************           STA  350
C                     CONSTRUCT EULER-TYPE PREDICTOR                STA  360
C                     CALCULATE APPROX. SOLN. AT TA+H               STA  370
C                     SEARCH FOR LAG CORRESPONDING TO               STA  380
C                     X(TA+H) JUST CALCULATED.                      STA  390
C                     APPROXIMATE X AT LAG VIA EULER                STA  400
C                    ************************************           STA  410
   20 CONTINUE                                                       STA  420
      H = H1                                                         STA  430
      ISWITC = 1                                                     STA  440
      IF (JRAY.NE.JEND) H = T(JRAY+1) - T(JRAY)                      STA  450
      DO 30 I=1,N                                                    STA  460
        L5 = I5 + I                                                  STA  470
        L6 = I6 + I                                                  STA  480
        W(L5) = X(I,JRAY)                                            STA  490
        W(L6) = XL(I,JRAY)                                           STA  500
   30 CONTINUE                                                       STA  510
      DO 40 I=1,N                                                    STA  520
        L1 = I1 + I                                                  STA  530
        W(L1) = F(T(JRAY),W(I5+1),W(I6+1),I)                         STA  540
   40 CONTINUE                                                       STA  550
      DO 50 I=1,N                                                    STA  560
        L5 = I5 + I                                                  STA  570
        L1 = I1 + I                                                  STA  580
        W(L5) = X(I,JRAY) + H*W(L1)                                  STA  590
C                    ************************************           STA  600
C                     IN EACH COMPONENT CALCULATE THE               STA  610
C                     CORRESPONDING LAG,W(I), FOR THE               STA  620
C                     EULER PREDICTED VALUE W(L5), AND              STA  630
C                     EXECUTE SEARCH.                               STA  640
C                    ************************************           STA  650
   50 CONTINUE                                                       STA  660
      DO 90 I=1,N                                                    STA  670
        W(I) = ALPHA(W(I5+1),T(JRAY)+H,I)                            STA  680
        J = JRAY                                                     STA  690
   60   TEST = W(I) - T(J)                                           STA  700
        IF (H1.LT.0.) TEST = -TEST                                   STA  710
        IF (TEST.GE.0.) GO TO 70                                     STA  720
        J = J - 1                                                    STA  730
        IF (J.EQ.0) GO TO 70                                         STA  740
        GO TO 60                                                     STA  750
C                                                                    STA  760
   70   CONTINUE                                                     STA  770
        L9 = I9 + I                                                  STA  780
        W(L9) = J                                                    STA  790
        IF (J.EQ.0) GO TO 90                                         STA  800
        DO 80 II=1,N                                                 STA  810
          L3 = I3 + II                                               STA  820
          L4 = I4 + II                                               STA  830
          W(L3) = X(II,J)                                            STA  840
          W(L4) = XL(II,J)                                           STA  850
   80   CONTINUE                                                     STA  860
   90 CONTINUE                                                       STA  870
      DO 100 I=1,N                                                   STA  880
        L9 = I9 + I                                                  STA  890
        J = W(L9)                                                    STA  900
        H = H1                                                       STA  910
        L6 = I6 + I                                                  STA  920
        IF (J.EQ.0) W(L6) = FINT(W(I),I)                             STA  930
```

```
      IF (J.EQ.0) GO TO 100                                              STA  940
      IF (J.NE.JEND) H = T(J+1) - T(J)                                   STA  950
C                 ************************************                   STA  960
C                  W(L6) IS THE SOLUTION AT ALPHA(W(I),T,I)              STA  970
C                 ************************************                   STA  980
      W(L6) = X(I,J) + (W(I)-T(J))*F(T(J),W(I3+1),W(I4+1),I)             STA  990
  100 CONTINUE                                                           STA 1000
C                 ************************************                   STA 1010
C                  REPEAT ENTIRE PROCEDURE ABOVE                         STA 1020
C                  USING EULER VALUE TO CALCULATE                        STA 1030
C                  HEUN-TYPE 2ND ORDER PREDICTOR                         STA 1040
C                 ************************************                   STA 1050
      DO 110 I=1,N                                                       STA 1060
        L2 = I2 + I                                                      STA 1070
        W(L2) = F(T(JRAY)+H,W(I5+1),W(I6+1),I)                           STA 1080
  110 CONTINUE                                                           STA 1090
      DO 120 I=1,N                                                       STA 1100
        L1 = I1 + I                                                      STA 1110
        L2 = I2 + I                                                      STA 1120
        L3 = I3 + I                                                      STA 1130
        L7 = I7 + I                                                      STA 1140
C                 ************************************                   STA 1150
C                  FOR EACH COMPONENT ESTIMATE THE                       STA 1160
C                  SOLUTION AT T+H AND T+H/2 AND STORE                   STA 1170
C                  ANSWERS IN W(L3) AND W(L7) RESP.                      STA 1180
C                 ************************************                   STA 1190
        W(L3) = X(I,JRAY) + H*W(L1) + .5*H*(W(L2)-W(L1))                 STA 1200
        W(L7) = X(I,JRAY) + .5*H*W(L1) + .125*H*(W(L2)-W(L1))            STA 1210
C                 ************************************                   STA 1220
C                  IN STATEMENTS 120 THRU 280 THE COR-                   STA 1230
C                  RESPONDING LAGS AND THE SOLUTION AT                   STA 1240
C                  THESE LAGS ARE COMPUTED TO 2ND ORDER                  STA 1250
C                  BY HEUN METHOD AND STORED IN W(L4) AND                STA 1260
C                  W(L8) RESPECTIVELY                                    STA 1270
C                 ************************************                   STA 1280
  120 CONTINUE                                                           STA 1290
      DO 130 I=1,N                                                       STA 1300
        W(I) = ALPHA(W(I3+1),T(JRAY)+H,I)                                STA 1310
  130 CONTINUE                                                           STA 1320
  140 DO 170 I=1,N                                                       STA 1330
        J = JRAY                                                         STA 1340
  150   TEST = W(I) - T(J)                                               STA 1350
        IF (H1.LT.0.) TEST = -TEST                                       STA 1360
        IF (TEST.GE.0.) GO TO 160                                        STA 1370
        J = J - 1                                                        STA 1380
        IF (J.EQ.0) GO TO 160                                            STA 1390
        GO TO 150                                                        STA 1400
C                                                                        STA 1410
  160   CONTINUE                                                         STA 1420
        L9 = I9 + I                                                      STA 1430
        W(L9) = J                                                        STA 1440
  170 CONTINUE                                                           STA 1450
      DO 290 I=1,N                                                       STA 1460
        L9 = I9 + I                                                      STA 1470
        J = W(L9)                                                        STA 1480
        H = H1                                                           STA 1490
        IF (J.EQ.0) GO TO 250                                            STA 1500
        IF (J.NE.JEND) H = T(J+1) - T(J)                                 STA 1510
        IF (J.EQ.JEND) GO TO 220                                         STA 1520
        DO 180 II=1,N                                                    STA 1530
          L5 = I5 + II                                                   STA 1540
          L6 = I6 + II                                                   STA 1550
          W(L5) = X(II,J)                                                STA 1560
          W(L6) = XL(II,J)                                               STA 1570
  180   CONTINUE                                                         STA 1580
        DO 190 II=1,N                                                    STA 1590
          L1 = I1 + II                                                   STA 1600
          W(L1) = F(T(J),W(I5+1),W(I6+1),II)                             STA 1610
  190   CONTINUE                                                         STA 1620
        DO 200 II=1,N                                                    STA 1630
          L5 = I5 + II                                                   STA 1640
          L6 = I6 + II                                                   STA 1650
          W(L5) = X(II,J+1)                                              STA 1660
          W(L6) = XL(II,J+1)                                             STA 1670
  200   CONTINUE                                                         STA 1680
        DO 210 II=1,N                                                    STA 1690
          L2 = I2 + II                                                   STA 1700
          W(L2) = F(T(J+1),W(I5+1),W(I6+1),II)                           STA 1710
  210   CONTINUE                                                         STA 1720
  220   GO TO (230, 240), ISWITC                                         STA 1730
  230   L1 = I1 + I                                                      STA 1740
        L2 = I2 + I                                                      STA 1750
        L4 = I4 + I                                                      STA 1760
        W(L4) = X(I,J) + (W(I)-T(J))*W(L1) + .5*(W(I)-T(J))**2*          STA 1770
     *   (W(L2)-W(L1))/H                                                 STA 1780
        GO TO 260                                                        STA 1790
C                                                                        STA 1800
  240   L1 = I1 + I                                                      STA 1810
        L2 = I2 + I                                                      STA 1820
        L8 = I8 + I                                                      STA 1830
```

```
      W(L8) = X(I,J) + (W(I)-T(J))*W(L1) + .5*(W(I)-T(J))**2*          STA 1840
     *   (W(L2)-W(L1))/H                                              STA 1850
  250 CONTINUE                                                        STA 1860
      L4 = I4 + I                                                     STA 1870
      IF (ISWITC.EQ.1) W(L4) = FINT(W(I),I)                           STA 1880
      L8 = I8 + I                                                     STA 1890
      IF (ISWITC.EQ.2) W(L8) = FINT(W(I),I)                           STA 1900
      GO TO 260                                                       STA 1910
C                                                                     STA 1920
  260 GO TO (270, 290), ISWITC                                        STA 1930
  270 DO 280 II=1,N                                                   STA 1940
        W(II) = ALPHA(W(I7+1),T(JRAY)+.5*H,II)                        STA 1950
  280 CONTINUE                                                        STA 1960
      ISWITC = ISWITC + 1                                             STA 1970
      GO TO 140                                                       STA 1980
C                                                                     STA 1990
C                 ***********************************               STA 2000
C                  USE HEUN PREDICTOR FOR 4TH ORDER                 STA 2010
C                  APPROX. SOLUTION.                                STA 2020
C                 ***********************************               STA 2030
  290 CONTINUE                                                        STA 2040
      H = H1                                                          STA 2050
      IF (JRAY.NE.JEND) H = T(JRAY+1) - T(JRAY)                       STA 2060
      D = F(T(JRAY)+H,W(I3+1),W(I4+1),JCOMP)                          STA 2070
      C = F(T(JRAY)+.5*H,W(I7+1),W(I8+1),JCOMP)                       STA 2080
      SS = B - T(JRAY)                                                STA 2090
      R = SS/H                                                        STA 2100
      DO 300 I=1,N                                                    STA 2110
        L5 = I5 + I                                                   STA 2120
        L6 = I6 + I                                                   STA 2130
        W(L5) = X(I,JRAY)                                             STA 2140
        W(L6) = XL(I,JRAY)                                            STA 2150
  300 CONTINUE                                                        STA 2160
      I = JCOMP                                                       STA 2170
      L1 = I1 + I                                                     STA 2180
      W(L1) = F(T(JRAY),W(I5+1),W(I6+1),JCOMP)                        STA 2190
C                 ***********************************               STA 2200
C                  RETURN 4TH ORDER APROXIMATION FOR                STA 2210
C                  USE IN F IN MERSON METHOD                        STA 2220
C                 ***********************************               STA 2230
      ANS = X(I,JRAY) + SS*(W(L1)+.5*R*(-3.*W(L1)+4.*C-D)+R*R/3.*     STA 2240
     * (2.*W(L1)-4.*C+2.*D))                                          STA 2250
      RETURN                                                          STA 2260
C                                                                     STA 2270
      END                                                             STA 2280
```

498-P 1- 0

ALGORITHM 498
Airy Functions Using Chebyshev Series Approximations

P.J. PRINCE

Teesside Polytechnic, Great Britain

Key Words and Phrases: Airy functions, Chebyshev series approximation, Chebyshev coefficients, asymptotic expansion, Taylor expansion
CR Categories: 5.12
Language: Fortran

DESCRIPTION

Purpose

This subroutine evaluates the Airy functions $\mathrm{Ai}(z)$ and $\mathrm{Bi}(z)$ and their respective derivatives Ai′ and Bi′, for all real values of z within computer capability, by means of Chebyshev series approximations.

Method

(a) For $z \leq -7$ the following asymptotic expansions are used (see [1, eqs. 10.4.60, 10.4.64, 10.4.62, and 10.4.67]):

$$\mathrm{Ai}(-z) \sim z^{-1/4}\{sf(z) - cg(z)\}, \qquad \mathrm{Ai}'(-z) \sim -z^{1/4}\{cp(z) + sq(z)\}$$

$$\mathrm{Bi}(-z) \sim z^{-1/4}\{cf(z) + sg(z)\}, \qquad \mathrm{Bi}'(-z) \sim z^{1/4}\{sp(z) - cq(z)\}$$

where $s = \sin(\zeta + \pi/4)$, $c = \cos(\zeta + \pi/4)$, $\zeta = 2/3\ z^{3/2}$, and f, g, p, and q are approximated by the following Chebyshev expansions:

$$f \simeq \sum_{r=0}^{m_1}{}' a_r T_r^*(t), \quad g \simeq \zeta^{-1} \sum_{r=0}^{m_2}{}' b_r T_r^*(t), \quad p \simeq \sum_{r=0}^{m_3}{}' c_r T_r^*(t), \quad q \simeq \zeta^{-1} \sum_{r=0}^{m_4}{}' d_r T_r^*(t)$$

where $t = -(7/z)^3$, the T_r^* are the shifted Chebyshev polynomials, and the a_r, b_r, c_r, and d_r are the prescribed Chebyshev coefficients.

(b) For $-7 < z \leq 0$ the functions are given by the following Taylor expansions (see [1, eqs. 10.4.2 and 10.4.3]):

$$\mathrm{Ai} = e_1 f(z) - e_2 g(z), \qquad \mathrm{Ai}' = e_1 f'(z) - e_2 g'(z)$$

$$\mathrm{Bi} = \sqrt{3}\{e_1 f(z) + e_2 g(z)\}, \qquad \mathrm{Bi}' = \sqrt{3}\{e_1 f'(z) + e_2 g'(z)\}$$

Received 2 August 1974 and 1 April 1975.

Author's address: Mathematics Department, Teesside Polytechnic, Middlesbrough, Cleveland, TS1 3BA, Great Britain.

where e_1 and e_2 are prescribed constants, and

$$f \simeq \sum_{r=0}^{m_1}{}' a_r T_r^*(t), \qquad g \simeq z \sum_{r=0}^{m_2}{}' b_r T_r^*(t)$$

$$f' \simeq z^2 \sum_{r=0}^{m_3}{}' c_r T_r^*(t), \qquad g' \simeq \sum_{r=0}^{m_4}{}' d_r T_r^*(t)$$

where $t = -(z/7)^3$.

(c) For $0 < z < 7$ the previous Taylor expansions do not allow the functions to be accurately calculated (unless using very high precision) at all points of the range because of the large fluctuation in magnitude of the functions over this range. To overcome this, the following Chebyshev approximations to the weighted functions are used:

$$\mathrm{Ai}\exp(1.75z) \simeq \sum_{r=0}^{m_1}{}' a_r T_r^*(t), \qquad \mathrm{Ai}'\exp(1.75z) \simeq \sum_{r=0}^{m_3}{}' c_r T_r^*(t)$$

$$\mathrm{Bi}\exp(-1.75z) \simeq \sum_{r=0}^{m_2}{}' b_r T_r^*(t), \qquad \mathrm{Bi}'\exp(-1.75z) \simeq \sum_{r=0}^{m_4}{}' d_r T_r^*(t)$$

where $t = z/7$.

(d) For $z \geq 7$ the following asymptotic approximations are used (see [1, eqs. 10.4.59, 10.4.63, 10.4.61, and 10.4.66]):

$$\mathrm{Ai}(z) \sim z^{-1/4}\exp(-\zeta) \sum_{r=0}^{m_1}{}' a_r T_r^*(t), \qquad \mathrm{Ai}'(z) \sim -z^{1/4}\exp(-\zeta) \sum_{r=0}^{m_3}{}' c_r T_r^*(t)$$

$$\mathrm{Bi}(z) \sim z^{-1/4}\exp(\zeta) \sum_{r=0}^{m_2}{}' b_r T_r^*(t), \qquad \mathrm{Bi}'(z) \sim z^{1/4}\exp(\zeta) \sum_{r=0}^{m_4}{}' d_r T_r^*(t)$$

where $t = (7/z)^{3/2}$.

In each case the degree of approximation, m_i, is such that the functions are determined accurate to at least nine significant figures if the function is greater than unity in magnitude and to nine decimal places if not. However, in case (a) the final accuracy is limited by the accuracy to which the sine and cosine of $\zeta + \pi/4$ can be computed. In the cases (a), (b), and (d) the Chebyshev coefficients were computed by rearrangement of the corresponding power series expansions using [3, p. 52, eq. 23]. In case (c) the coefficients were computed by numerical integration based on [3, p. 65, eq. 1], where the required Airy function values were computed in high precision using the Taylor expansions. In all four cases use is made of the Rice algorithm for summing the Chebyshev series (see [3, p. 57, eq. 50]).

Program

The subroutine is named AIRY and is composed of the four subroutines COEF1, COEF2, COEF3, and COEF4 corresponding, respectively, to the previous cases (a), (b), (c), and (d). The calling statement is

CALL AIRY (Z, NFUNC, ISCAL, AI, BI, AID, BID)

where the input variables Z, NFUNC, and ISCAL are defined as follows:

Z the prescribed argument

NFUNC an indicator for which of the functions are to be calculated, as follows:

NFUNC { negative, for Ai' and Bi' only;
0, for all four functions;
positive, for Ai and Bi only

ISCAL an indicator for whether or not in case (d) the following scaling is required:

$$(\mathrm{Ai}, \mathrm{Ai}') = (\mathrm{Ai}, \mathrm{Ai}') \exp(\zeta);$$
$$(\mathrm{Bi}, \mathrm{Bi}') = (\mathrm{Bi}, \mathrm{Bi}') \exp(-\zeta);$$

if ISCAL = 0 there is no scaling; otherwise, scaling as above.

The output variables will contain the respective values of Ai, Bi, Ai′, and Bi′. After the CALL statement the input parameters are left unchanged.

The relevant prescribed Chebyshev coefficients are stored in arrays A, B, C, and D corresponding, respectively, to a_r, b_r, c_r, and d_r in the previous analysis. Subroutine CHEB evaluates the sum of the required Chebyshev series given the argument $x (0 \leq x \leq 1)$ and the corresponding N Chebyshev coefficients $a_0, a_1, \ldots, a_{N-1}$ stored as $A(I)$, $I = 1, N$.

Test Results

Tests have been carried out on an ICL 1905E computer (11S floating-point arithmetic). The Wronskian relationship $W = \mathrm{AiBi}' - \mathrm{Ai}'\mathrm{Bi} = 1/\pi$ has been used to check on the results, as have the tables in [1] and the results obtained from a high precision calculation based on the Taylor expansions.

Tests have also been carried out for comparison with ACM Algorithm 301 [2], which gives at least eight-figure accuracy; the present algorithm is comparable in time for setting up large tables of the functions, and is certainly more efficient when only a small number of function evaluations are required outside the asymptotic ranges. For shorter word machines the user may truncate the given Chebyshev coefficients. The routine has run successfully on a Data General Corporation Nova 840 (7S floating-point arithmetic), with at least five-figure accuracy obtained.

REFERENCES

1. Abramowitz, M., and Stegun, I.A. *Handbook of Mathematical Functions.* Dover Publications, New York, 1965, pp. 446, 448–449, 475–477.
2. Bond, G., and Pitteway, M.L.V. Algorithm 301, Airy function. *Comm. ACM 10,* 5 (May 1967), 291–293.
3. Fox, L., and Parker, I.B. *Chebyshev Polynomials in Numerical Analysis.* Oxford U. Press, London, 1968, pp. 48–58, 65–68.

ALGORITHM

```
      SUBROUTINE AIRY(Z, NFUNC, ISCAL, AI, BI, AID, BID)                AIR   10
C P.J.PRINCE, MATHEMATICS DEPARTMENT, TEESSIDE POLYTECHNIC,             AIR   20
C MIDDLESBROUGH, CLEVELAND TS1 3BA, GREAT BRITAIN.                      AIR   30
C JANUARY  9, 1975 .                                                    AIR   40
C REFERENCES -                                                          AIR   50
C (1) - ABRAMOWITZ,M. AND STEGUN,I.A. HANDBOOK OF                       AIR   60
C       MATHEMATICAL FUNCTIONS. DOVER PUBLICATIONS INC.,                AIR   70
C       NEW YORK, 1965 PAGES 446,448-449,475-477 .                      AIR   80
C (2) - FOX,L. AND PARKER,I.B. CHEBYSHEV POLYNOMIALS IN                 AIR   90
C       NUMERICAL ANALYSIS. OXFORD UNIVERSITY PRESS,1968                AIR  100
C       PAGES 48-58,65-68 .                                             AIR  110
C (3) - BOND,GILLIAN AND PITTEWAY,M.L.V. ALGORITHM 301,                 AIR  120
C       AIRY FUNCTION(S20). C.A.C.M. VOL. 10,NO. 5, MAY 1967            AIR  130
C THIS SUBROUTINE EVALUATES THE AIRY FUNCTIONS AND THEIR                AIR  140
C DERIVATIVES FOR ANY REAL ARGUMENT WITHIN COMPUTER                     AIR  150
C CAPABILITY, BY MEANS OF CHEBYSHEV SERIES APPROXIMATIONS.              AIR  160
C Z          -   THE ARGUMENT FOR WHICH THE FUNCTIONS ARE               AIR  170
C                TO BE EVALUATED.                                       AIR  180
C AI,BI      -   THE AIRY FUNCTIONS.                                    AIR  190
C AID,BID    -   THEIR RESPECTIVE DERIVATIVES.                         AIR  200
C NFUNC      -   INDICATES WHICH OF THE FUNCTIONS ARE TO BE            AIR  210
C                CALCULATED.                                            AIR  220
C ISCAL      -   INDICATES WHETHER OR NOT SCALING IS                   AIR  230
C                REQUIRED.                                              AIR  240
C FOUR RANGES OF ARGUMENT Z ARE CONSIDERED -                            AIR  250
C (A)  Z .LE. -7.0                                                      AIR  260
C (B)  -7.0 .LT. Z .LE. 0.0                                             AIR  270
C (C)  0.0 .LT. Z .LT. 7.0                                              AIR  280
C (D)  Z .GE. 7.0                                                       AIR  290
C THE SUBROUTINES COEF1, COEF2, COEF3 AND COEF4 WHICH ARE               AIR  300
C CALLED BY THIS SUBROUTINE CORRESPOND RESPECTIVELY TO                  AIR  310
C THESE FOUR RANGES.                                                    AIR  320
C THE FUNCTIONS CALCULATED ARE AS FOLLOWS -                             AIR  330
C IF NFUNC IS NEGATIVE  AID,BID.                                        AIR  340
```

```
C IF NFUNC = 0  ALL FOUR.                                                AIR  350
C IF NFUNC IS POSITIVE  AI,BI.                                           AIR  360
C ISCAL INDICATES WHETHER OR NOT IN RANGE (D) THE FOLLOWING              AIR  370
C SCALING IS TO BE CARRIED OUT -                                         AIR  380
C (AI,AID) = (AI,AID)*EXP(ZETA), (BI,BID) = (BI,BID)*EXP(-ZETA)          AIR  390
C WHERE ZETA = 2.0/3.0*Z**1.5                                            AIR  400
C IF ISCAL = 0 THEN THERE IS NO SCALING OTHERWISE SCALING                AIR  410
C AS ABOVE.                                                              AIR  420
C IN (A) THE FINAL ACCURACY IS HEAVILY DEPENDENT ON THE                  AIR  430
C ACCURACY TO WHICH THE SINE AND COSINE OF                               AIR  440
C ANG = 2.0/3.0*(-Z)**1.5+PI/4 CAN BE COMPUTED.                          AIR  450
C IF ANGLM .LT. ANG .LE. ANGUP THEN SOME ACCURACY MAY BE                 AIR  460
C LOST. A NON FATAL ERROR WARNING IS GIVEN.                              AIR  470
C IF ANG .GT. ANGUP A NON FATAL ERROR WARNING IS GIVEN AND               AIR  480
C THE FOUR FUNCTIONS ARE ASSIGNED THE VALUE ZERO.                        AIR  490
C IN (D) IF ISCAL = 0 OVERFLOW IN EXP WILL OCCUR IF                      AIR  500
C Z .GE. ZLIM. IN THIS CASE A NON FATAL ERROR WARNING IS                 AIR  510
C GIVEN AND THE FOLLOWING VALUES ARE ASSIGNED -                          AIR  520
C AI = AID = 0.0                                                         AIR  530
C BI = BID = EXP(ZLIM).                                                  AIR  540
C IF ISCAL .NE. 0 OVERFLOW WILL OCCUR IF                                 AIR  550
C Z .GT. ZUP = ZMAX**(2.0/3.0) WHERE ZMAX IS THE MAXIMUM                 AIR  560
C REPRESENTABLE NUMBER ON THE COMPUTER. IN THIS CASE A                   AIR  570
C NON FATAL ERROR WARNING IS GIVEN AND THE FOUR FUNCTIONS                AIR  580
C ARE ASSIGNED THE VALUE ZERO.                                           AIR  590
C NOUT IS THE OUTPUT CHANNEL USED.                                       AIR  600
C THE MACHINE DEPENDENT CONSTANTS (VALUES BEING ASSIGNED BY              AIR  610
C APPROPRIATE DATA STATEMENTS) ARE -                                     AIR  620
C IN COEF1  -  ANGLM,ANGUP AND NOUT.                                     AIR  630
C IN COEF4  -  ZLIM,ZUP AND NOUT.                                        AIR  640
C FOR I.C.L. 1905E(11S FLOATING POINT ARITHMETIC) -                      AIR  650
  ANGLM = 250.0, ANGUP = 1.0E10, NOUT = 2, ZLIM = 175.0,                 AIR  660
  ZUP = 1.0E50, ZMAX = 5.6E76                                            AIR  670
      IF (Z) 10, 10, 40                                                  AIR  680
   10 IF (Z+7.0) 20, 20, 30                                              AIR  690
   20 CALL COEF1(Z, NFUNC, AI, BI, AID, BID)                             AIR  700
      RETURN                                                             AIR  710
   30 CALL COEF2(Z, NFUNC, AI, BI, AID, BID)                             AIR  720
      RETURN                                                             AIR  730
   40 IF (Z-7.0) 50, 60, 60                                              AIR  740
   50 CALL COEF3(Z, NFUNC, AI, BI, AID, BID)                             AIR  750
      RETURN                                                             AIR  760
   60 CALL COEF4(Z, NFUNC, ISCAL, AI, BI, AID, BID)                      AIR  770
      RETURN                                                             AIR  780
      END                                                                AIR  790



      SUBROUTINE COEF1(Z, NFUNC, AI, BI, AID, BID)                       COE   10
C THIS SUBROUTINE EVALUATES THE FOUR FUNCTIONS IN THE CASE               COE   20
C WHEN Z .LE. -7.0 USING CHEBYSHEV SERIES APPROXIMATIONS TO              COE   30
C THE CORRESPONDING ASYMPTOTIC EXPANSIONS.                               COE   40
      DIMENSION A(5), B(5), C(5), D(5)                                   COE   50
      DATA A(1), A(2), A(3), A(4), A(5) /1.1282427601,                   COE   60
     * -0.6803534E-04,0.16687E-06,-0.128E-08,0.2E-10/                    COE   70
      DATA B(1), B(2), B(3), B(4), B(5) /0.7822108673E-01,               COE   80
     * -0.6895649E-04,0.32857E-06,-0.370E-08,0.7E-10/                    COE   90
      DATA C(1), C(2), C(3), C(4), C(5) /1.1285404716,0.8046925E-04,     COE  100
     * -0.18161E-06,0.135E-08,-0.2E-10/                                  COE  110
      DATA D(1), D(2), D(3), D(4), D(5) /-0.10954855184,                 COE  120
     * 0.7713350E-04,-0.35168E-06,0.388E-08,-0.7E-10/                    COE  130
      DATA PI4 /0.78539816340/                                           COE  140
      DATA ANGLM, ANGUP /250.0,1.0E10/                                   COE  150
      DATA NOUT /2/                                                      COE  160
      X = -Z                                                             COE  170
      SX = SQRT(X)                                                       COE  180
      Y = X*SX                                                           COE  190
      ZETA = 0.66666666667*Y                                             COE  200
      Z4 = SQRT(SX)                                                      COE  210
      ANG = ZETA + PI4                                                   COE  220
C TEST ARGUMENT SIZE FOR SIN AND COS.                                    COE  230
      IF (ANGLM-ANG) 10, 40, 40                                          COE  240
   10 IF (ANGUP-ANG) 20, 20, 30                                          COE  250
   20 WRITE (NOUT,99999) Z, ANG                                          COE  260
      AI = 0.0                                                           COE  270
      BI = 0.0                                                           COE  280
      AID = 0.0                                                          COE  290
      BID = 0.0                                                          COE  300
      RETURN                                                             COE  310
   30 WRITE (NOUT,99998) Z, ANG                                          COE  320
   40 SN = SIN(ANG)                                                      COE  330
      CN = COS(ANG)                                                      COE  340
      ZETA1 = 1.0/ZETA                                                   COE  350
      X = 7.0/X                                                          COE  360
      X = X*X*X                                                          COE  370
C ARGUMENT SCALED TO RANGE (0,1).                                        COE  380
C EVALUATE THE RELEVANT SERIES.                                          COE  390
      IF (NFUNC.LT.0) GO TO 50                                           COE  400
      CALL CHEB(X, 5, A, FA)                                             COE  410
      CALL CHEB(X, 5, B, FB)                                             COE  420
      Z4I = 1.0/Z4                                                       COE  430
```

```
      FB = FB*ZETAI                                                     COE  440
      AI = Z4I*(SN*FA-CN*FB)                                            COE  450
      BI = Z4I*(CN*FA+SN*FB)                                            COE  460
      IF (NFUNC.GT.0) RETURN                                            COE  470
   50 CALL CHEB(X, 5, C, FC)                                            COE  480
      CALL CHEB(X, 5, D, FD)                                            COE  490
      FD = FD*ZETAI                                                     COE  500
      AID = -Z4*(CN*FC+SN*FD)                                           COE  510
      BID = Z4*(SN*FC-CN*FD)                                            COE  520
      RETURN                                                            COE  530
99999 FORMAT (//38X, 3H***//5X, 28HANGLE OUTSIDE MACHINE RANGE,,        COE  540
     * 38H THE FOUR FUNCTIONS HAVE BEEN ASSIGNED/5X, 11HTHE VALUE Z,     COE  550
     * 4HERO.//5X, 4HZ = , E20.8, 10H  ANGLE = , E20.8//38X, 3H***//     COE  560
     * )                                                                 COE  570
99998 FORMAT (//38X, 3H***//5X, 25HSOME ACCURACY MAY BE LOST,            COE  580
     * 16H IN SIN AND COS.//5X, 4HZ = , E20.8, 10H  ANGLE = ,           COE  590
     * E20.8//38X, 3H***//)                                             COE  600
      END                                                               COE  610


      SUBROUTINE COEF2(Z, NFUNC, AI, BI, AID, BID)                      COE   10
C THIS SUBROUTINE EVALUATES THE FOUR FUNCTIONS IN THE CASE              COE   20
C WHEN -7.0 .LT. Z .LE. 0.0 USING CHEBYSHEV SERIES                      COE   30
C APPROXIMATIONS TO THE CORRESPONDING TAYLOR EXPANSIONS.                COE   40
      DIMENSION A(17), B(16), C(16), D(17)                              COE   50
      DATA A(1), A(2), A(3), A(4), A(5), A(6), A(7), A(8), A(9),        COE   60
     * A(10), A(11), A(12), A(13), A(14), A(15), A(16), A(17)           COE   70
     * /0.11535880704,0.6542816649E-01,0.26091774326,0.21959346500,     COE   80
     * 0.12476382168,-0.43476292594,0.28357718605,-0.9751797082E-01,    COE   90
     * 0.2182551823E-01,-0.350454097E-02,0.42778312E-03,                COE  100
     * -0.4127564E-04,0.323880E-05,-0.21123E-06,0.1165E-07,             COE  110
     * -0.55E-09,0.2E-10/                                               COE  120
      DATA B(1), B(2), B(3), B(4), B(5), B(6), B(7), B(8), B(9),        COE  130
     * B(10), B(11), B(12), B(13), B(14), B(15), B(16)                  COE  140
     * /0.10888288487,-0.17511655051,0.13887871101,-0.11469998998,      COE  150
     * 0.22377807641,-0.18546243714,0.8063565186E-01,                   COE  160
     * -0.2208847864E-01,0.422444527E-02,-0.60131028E-03,               COE  170
     * 0.6653890E-04,-0.590842E-05,0.43131E-06,-0.2638E-07,             COE  180
     * 0.137E-08,-0.6E-10/                                              COE  190
      DATA C(1), C(2), C(3), C(4), C(5), C(6), C(7), C(8), C(9),        COE  200
     * C(10), C(11), C(12), C(13), C(14), C(15), C(16)                  COE  210
     * /0.7571648463E-01,-0.10150232871,0.7800551669E-01,               COE  220
     * -0.8324569361E-01,0.10105322731,-0.6578603344E-01,               COE  230
     * 0.2500140353E-01,-0.625962704E-02,0.111945149E-02,               COE  240
     * -0.15102718E-03,0.1598086E-04,-0.136545E-05,0.9636E-07,          COE  250
     * -0.572E-08,0.29E-09,-0.1E-10/                                    COE  260
      DATA D(1), D(2), D(3), D(4), D(5), D(6), D(7), D(8), D(9),        COE  270
     * D(10), D(11), D(12), D(13), D(14), D(15), D(16), D(17)           COE  280
     * /0.61603048107,0.85738069722,0.86345334421,0.80890228699,        COE  290
     * -0.50565665369,-0.81829752697,0.77829538563,-0.31201242692,      COE  300
     * 0.7677186198E-01,-0.1320520264E-01,0.170185698E-02,              COE  310
     * -0.17177956E-03,0.1401068E-04,-0.94532E-06,0.5374E-07,           COE  320
     * -0.261E-08,0.11E-09/                                             COE  330
      DATA E1, E2, ROOT3 /0.35502805389,0.25881940379,1.7320508076/     COE  340
      X = -0.29154518950E-02*Z*Z*Z                                      COE  350
C ARGUMENT SCALED TO RANGE (0,1).                                       COE  360
C EVALUATE THE RELEVANT SERIES.                                         COE  370
      IF (NFUNC.LT.0) GO TO 10                                          COE  380
      CALL CHEB(X, 17, A, FA)                                           COE  390
      CALL CHEB(X, 16, B, FB)                                           COE  400
      FA = E1*FA                                                        COE  410
      FB = E2*Z*FB                                                      COE  420
      AI = FA - FB                                                      COE  430
      BI = ROOT3*(FA+FB)                                                COE  440
      IF (NFUNC.GT.0) RETURN                                            COE  450
   10 CALL CHEB(X, 16, C, FC)                                           COE  460
      CALL CHEB(X, 17, D, FD)                                           COE  470
      FC = E1*Z*Z*FC                                                    COE  480
      FD = E2*FD                                                        COE  490
      AID = FC - FD                                                     COE  500
      BID = ROOT3*(FC+FD)                                               COE  510
      RETURN                                                            COE  520
      END                                                               COE  530


      SUBROUTINE COEF3(Z, NFUNC, AI, BI, AID, BID)                      COE   10
C THIS SUBROUTINE EVALUATES THE FOUR FUNCTIONS IN THE CASE              COE   20
C WHEN 0.0 .LT. Z .LT. 7.0 USING CHEBYSHEV SERIES                       COE   30
C APPROXIMATIONS TO THE CORRESPONDING WEIGHTED FUNCTIONS.               COE   40
      DIMENSION A(20), B(25), C(22), D(24)                              COE   50
      DATA A(1), A(2), A(3), A(4), A(5), A(6), A(7), A(8), A(9),        COE   60
     * A(10), A(11), A(12), A(13), A(14), A(15), A(16), A(17),         COE   70
     * A(18), A(19), A(20) /1.2974695794,-0.20230907821,                COE   80
     * -0.45786169516,0.12953331987,0.6983827954E-01,                   COE   90
     * -0.3005148746E-01,-0.493036334E-02,0.390425474E-02,              COE  100
     * -0.1546539E-04,-0.32193999E-03,0.3812734E-04,0.1714935E-04,      COE  110
     * -0.416096E-05,-0.50623E-06,0.26346E-06,-0.281E-08,               COE  120
     * -0.1122E-07,0.120E-08,0.31E-09,-0.7E-10/                         COE  130
      DATA B(1), B(2), B(3), B(4), B(5), B(6), B(7), B(8), B(9),        COE  140
```

```
     * B(10), B(11), B(12), B(13), B(14), B(15), B(16), B(17),          COE  150
     * B(18), B(19), B(20), B(21), B(22), B(23), B(24), B(25)           COE  160
     * /0.47839902387,-0.6881732880E-01,0.20938146768,                  COE  170
     * -0.3988095886E-01,0.4758441683E-01,-0.812296149E-02,             COE  180
     * 0.4628459l3E-02,0.70010098E-03,-0.75611274E-03,                  COE  190
     * 0.68958657E-03,-0.33621865E-03,0.14501669E-03,-0.4766359E-04,    COE  200
     * 0.1251965E-04,-0.193012E-05,-0.19032E-06,0.29390E-06,            COE  210
     * -0.13436E-06,0.4231E-07,-0.967E-08,0.135E-08,0.7E-10,            COE  220
     * -0.12E-09,0.4E-10,-0.1E-10/                                      COE  230
      DATA C(1), C(2), C(3), C(4), C(5), C(6), C(7), C(8), C(9),       COE  240
     * C(10), C(11), C(12), C(13), C(14), C(15), C(16), C(17),         COE  250
     * C(18), C(19), C(20), C(21), C(22) /-2.2359158747,                COE  260
     * -0.2638272392E-01,0.95151904332,-0.8383641182E-01,               COE  270
     * -0.19401303219,0.3580664778E-01,0.2269348562E-01,                COE  280
     * -0.671179820E-02,-0.152460473E-02,0.75474150E-03,                COE  290
     * 0.3729934E-04,-0.5653536E-04,0.350796E-05,0.289418E-05,          COE  300
     * -0.47423E-06,-0.9449E-07,0.3054E-07,0.109E-08,-0.130E-08,        COE  310
     * 0.8E-10,0.4E-10,-0.1E-10/                                        COE  320
      DATA D(1), D(2), D(3), D(4), D(5), D(6), D(7), D(8), D(9),       COE  330
     * D(10), D(11), D(12), D(13), D(14), D(15), D(16), D(17),         COE  340
     * D(18), D(19), D(20), D(21), D(22), D(23), D(24)                  COE  350
     * /0.71364662990,0.23777925892,0.28219009446,0.4912480040E-01,     COE  360
     * 0.6741261353E-01,-0.406308553E-02,0.1544814895E-01,              COE  370
     * -0.449172894E-02,0.322474426E-02,-0.105361380E-02,               COE  380
     * 0.41311371E-03,-0.8536169E-04,0.655166E-05,0.960458E-05,         COE  390
     * -0.641792E-05,0.280308E-05,-0.89454E-06,0.21392E-06,             COE  400
     * -0.2958E-07,-0.309E-08,0.376E-08,-0.148E-08,0.39E-09,            COE  410
     * -0.7E-10/                                                        COE  420
      X = 0.142857142860*Z                                             COE  430
      EX = EXP(1.75*Z)                                                 COE  440
      EY = 1.0/EX                                                      COE  450
C ARGUMENT SCALED TO RANGE (0,1).                                      COE  460
C EVALUATE THE RELEVANT SERIES.                                        COE  470
      IF (NFUNC.LT.0) GO TO 10                                         COE  480
      CALL CHEB(X, 20, A, AI)                                          COE  490
      CALL CHEB(X, 25, B, BI)                                          COE  500
C MULTIPLY BY APPROPRIATE WEIGHTING FACTORS.                           COE  510
      AI = AI*EY                                                       COE  520
      BI = BI*EX                                                       COE  530
      IF (NFUNC.GT.0) RETURN                                           COE  540
   10 CALL CHEB(X, 22, C, AID)                                         COE  550
      CALL CHEB(X, 24, D, BID)                                         COE  560
C MULTIPLY BY APPROPRIATE WEIGHTING FACTORS.                           COE  570
      AID = AID*EY                                                     COE  580
      BID = BID*EX                                                     COE  590
      RETURN                                                           COE  600
      END                                                              COE  610



      SUBROUTINE COEF4(Z, NFUNC, ISCAL, AI, BI, AID, BID)              COE   10
C THIS SUBROUTINE EVALUATES THE FOUR FUNCTIONS (SCALED                 COE   20
C UNLESS ISCAL = 0) IN THE CASE WHEN Z .GE. 7.0 USING                  COE   30
C CHEBYSHEV SERIES APPROXIMATIONS TO THE CORRESPONDING                 COE   40
C ASYMPTOTIC EXPANSIONS.                                               COE   50
      DIMENSION A(7), B(7), C(7), D(7)                                 COE   60
      DATA A(1), A(2), A(3), A(4), A(5), A(6), A(7)                    COE   70
     * /0.56265126169,-0.76136219E-03,0.765252E-05,-0.14228E-06,       COE   80
     * 0.380E-08,-0.13E-09,0.1E-10/                                    COE   90
      DATA B(1), B(2), B(3), B(4), B(5), B(6), B(7)                    COE  100
     * /1.1316635302,0.166141673E-02,0.1968882E-04,0.47047E-06,        COE  110
     * 0.1769E-07,0.94E-09,0.6E-10/                                    COE  120
      DATA C(1), C(2), C(3), C(4), C(5), C(6), C(7)                    COE  130
     * /0.56635357764,0.107273242E-02,-0.910034E-05,0.15998E-06,       COE  140
     * -0.415E-08,0.14E-09,-0.1E-10/                                   COE  150
      DATA D(1), D(2), D(3), D(4), D(5), D(6), D(7)                    COE  160
     * /1.1238058432,-0.230925296E-02,-0.2309457E-04,-0.52171E-06.     COE  170
     * -0.1907E-07,-0.100E-08,-0.7E-10/                                COE  180
      DATA ZLIM, ZUP /175.0,1.0E50/                                    COE  190
      DATA NOUT /2/                                                    COE  200
C TEST FOR OVERFLOW IN Z**1.5 .                                        COE  210
      IF (ZUP-Z) 10, 20, 20                                            COE  220
   10 WRITE (NOUT,99999) Z                                             COE  230
      AI = 0.0                                                         COE  240
      BI = 0.0                                                         COE  250
      AID = 0.0                                                        COE  260
      BID = 0.0                                                        COE  270
      RETURN                                                           COE  280
   20 SZ = SQRT(Z)                                                     COE  290
      Y = Z*SZ                                                         COE  300
      Z4 = SQRT(SZ)                                                    COE  310
      IF (ISCAL) 30, 40, 30                                            COE  320
   30 EX = 1.0                                                         COE  330
      EY = 1.0                                                         COE  340
      GO TO 70                                                         COE  350
   40 ZETA = 0.66666666667*Y                                           COE  360
C TEST FOR OVERFLOW IN EXP                                             COE  370
      IF (ZLIM-ZETA) 50, 60, 60                                        COE  380
   50 WRITE (NOUT,99998) Z, ZETA                                       COE  390
      AI = 0.0                                                         COE  400
      AID = 0.0                                                        COE  410
```

```
      BI = EXP(ZLIM)                                                    COE  420
      BID = BI                                                          COE  430
      RETURN                                                            COE  440
   60 EX = EXP(ZETA)                                                    COE  450
      EY = 1.0/EX                                                       COE  460
   70 X = 18.520259177/Y                                                COE  470
C ARGUMENT SCALED TO RANGE (0,1).                                       COE  480
C EVALUATE THE RELEVANT SERIES.                                         COE  490
      IF (NFUNC.LT.0) GO TO 80                                          COE  500
      CALL CHEB(X, 7, A, AI)                                            COE  510
      CALL CHEB(X, 7, B, BI)                                            COE  520
      Z4I = 1.0/Z4                                                      COE  530
      AI = Z4I*AI*EY                                                    COE  540
      BI = Z4I*BI*EX                                                    COE  550
      IF (NFUNC.GT.0) RETURN                                            COE  560
   80 CALL CHEB(X, 7, C, AID)                                           COE  570
      CALL CHEB(X, 7, D, BID)                                           COE  580
      AID = -Z4*AID*EY                                                  COE  590
      BID = Z4*BID*EX                                                   COE  600
      RETURN                                                            COE  610
99999 FORMAT (//38X, 3H***//5X, 18HOVERFLOW IN Z**1.5//5X, 4HZ = ,      COE  620
     * E20.8//5X, 41HTHE FOUR FUNCTIONS HAVE BEEN ASSIGNED THE,         COE  630
     * 12H VALUE ZERO.//38X, 3H***//)                                   COE  640
99998 FORMAT (//38X, 3H***//5X, 21HOVERFLOW IN EXP(ZETA)//5X,           COE  650
     * 4HZ = , E20.8, 9H  ZETA = , E20.8//5X, 18HTHE FOLLOWING VALU,    COE  660
     * 23HES HAVE BEEN ASSIGNED -//5X, 25HAI = AID = 0.0, BI = BID ,    COE  670
     * 11H= EXP(ZLIM)//38X, 3H***//)                                    COE  680
      END                                                               COE  690


      SUBROUTINE CHEB(X, N, A, F)                                       CHE   10
C THIS SUBROUTINE APPLIES THE RICE ALGORITHM TO THE SUM                 CHE   20
C F(X) = SIGMA-PRIME (R=1 TO N) A(R)*TSTAR-SUB-(R-1)(X)                 CHE   30
C WHERE THE TSTAR-SUB-R(X) ARE THE SHIFTED CHEBYSHEV                    CHE   40
C POLYNOMIALS.                                                          CHE   50
C X    -    THE ARGUMENT FOR WHICH THE SERIES IS TO BE                  CHE   60
C           EVALUATED.  (0.0 .LE. X .LE. 1.0).                         CHE   70
C N    -    THE NUMBER OF TERMS OF THE SERIES TO BE SUMMED.             CHE   80
C A(R) -    THE CHEBYSHEV COEFFICIENTS FOR THE SERIES.                  CHE   90
C F    -    THE COMPUTED SUM OF THE SERIES.                             CHE  100
      DIMENSION A(25)                                                   CHE  110
      B = 0.0                                                           CHE  120
      D = A(N)                                                          CHE  130
      U = X + X - 1.0                                                   CHE  140
      Y = U + U                                                         CHE  150
      J = N - 1                                                         CHE  160
      DO 10 I=3,N                                                       CHE  170
        C = B                                                           CHE  180
        B = D                                                           CHE  190
        D = Y*B - C + A(J)                                              CHE  200
        J = J - 1                                                       CHE  210
   10 CONTINUE                                                          CHE  220
      F = U*D - B + 0.5*A(1)                                            CHE  230
      RETURN                                                            CHE  240
      END                                                               CHE  250
```

REMARK ON ALGORITHM 498

Airy Functions Using Chebyshev Series Approximations [P.J. Price, *ACM Trans. Math. Softw. 1*, 4 (Dec. 1975), 372-379]

M. Razaz and J.L. Schonfelder [Received 19 March 1980, accepted 11 June 1980]

The Computer Centre, University of Birmingham, Elms Road, Birmingham, England.

Algorithm 498 presents a method for approximating the Airy functions and their first derivatives based on Chebyshev expansions. The method employed and the actual Chebyshev expansions presented give adequate routines for accuracies limited to about 8D (absolute in the negative region and relative in the positive). Such limited accuracy is adequate for single precision on 32- or 36-bit machines, that is, working precisions of 6D to 8D, approximately. Machines currently in common use regularly employ floating-point precisions of 11, 15, 17, and 18D. To cover this accuracy range and to deal with possible double-precision implementations, we have extended the accuracy capabilities of this basic algorithm to 30D

and new Chebyshev expansions have been generated to that precision [1]. This necessitated some modifications to the basic method of Algorithm 498. The most significant of such modifications was the need to introduce an additional range subdivision for the positive region.

Following as closely as possible the notation of Algorithm 498, we use for regions (a) and (b) entirely similar expansions, with the essential differences that the break point is chosen at -5 rather than -7 and we absorb the constants c_1 and c_2 of region (b) into the Chebyshev series. The former change is merely to improve the balance of the approximations for the important accuracy range 15D to 20D and to speed up the routines for the important argument range that includes the first few oscillations of the functions.

In order to increase the precision in the positive region it becomes necessary to introduce an additional region (c′), analogous to region (c). For the latter region we employ

$$Ai(z)\exp(+\alpha z) = \sum_{r=0}' a_r T_r^*(t), \qquad Ai'(z)\exp(+\beta z) = \sum_{r=0}' c_r T_r^*(t)$$

$$Bi(z)\exp(-\beta z) = \sum_{r=0}' b_r T_r^*(t), \qquad Bi'(z)\exp(-\alpha z) = \sum_{r=0}' d_r T_r^*(t)$$

with $0 < z < 4.5$, $t = z/4.5$, $\alpha = 1.5$, and $\beta = 1.375$. The prime means that the term with suffix $r = 0$ in each expansion is halved. The constants α and β were chosen to approximately minimize the variation of the function being expanded as a polynomial, this being required to give good relative error control for the resulting approximations [2]. These constants were also chosen to be exactly representable on any existing machine, thus aiding portability and preserving accuracy.

For the additional region (c′) we use

$$Ai(z)\exp(+\alpha z) = \sum_{r=0}' a_r T_r^*(t), \qquad Ai'(z)\exp(+\alpha z) = \sum_{r=0}' c_r T_r^*(t)$$

$$Bi(z)\exp(-\beta z) = \sum_{r=0}' b_r T_r^*(t), \qquad Bi'(z)\exp(-\beta z) = \sum_{r=0}' d_r T_r^*(t)$$

with $4.5 < z < 9$, $t = (z - 4.5)/4.5$, $\alpha = 2.5$, and $\beta = 2.625$.

For region (d) we use expansions as in Algorithm 498, except that the break point of 9 rather than 7 is used. This allows both slightly faster convergence for the expansions and a more efficient and accurate calculation of t to be performed. The parameter t can now be calculated as

$$t = (9/z)^{3/2} = 27/(z)^{3/2} = 18/\zeta.$$

Thus we avoid the necessity of calculating a 3/2 power more than once and/or of carrying a further nonexact constant $(7)^{3/2}$.

The modifications to the code of Algorithm 498 to employ these new expansions for obtaining a higher precision routine are fairly straightforward and the actual expansions have been published elsewhere [1]. Using these expansions the authors have produced routines that have been included in the S17 chapter of the NAG library [3]. The routines have been tested and have been shown to provide approximations accurate to within the errors expected for functions of this type for machines of precisions up to 18D, the errors being of the order of the machine precision except where losses are unavoidable. For instance, for large arguments where the errors are dominated by the imprecisions of sine, cosine, and exponential routines.

REFERENCES

1. RAZAZ, M., AND SCHONFELDER, J.L. High precision Chebyshev expansions for Airy functions and their derivatives. Submitted to *IMA J. Numer. Anal.*, to appear.
2. RAZAZ, M., AND SCHONFELDER, J.L. Error control with polynomial approximations. *IMA J. Numer. Anal. 1* (1980), 105–114.
3. SCHONFELDER, J.L. The production and testing of special function routines for a multi-machine library. *Softw. Pract. Exper. 6* (1976), 71–82.

ALGORITHM 499
An Efficient Scanning Technique [Z]

W. KINSNER and E. DELLA TORRE

McMaster University, Canada

Key Words and Phrases: array scanning, pattern recognition, partial differential equations, finite-differences, Laplace's equation
CR Categories: 4.22, 5.14, 5.17, 8.2
Language: Fortran

DESCRIPTION

1. Introduction

Scanning an array refers to selecting all the elements of the array in a particular order. We shall consider two scanning techniques: a raster scan and a new contour scan. The raster scan selects the elements of a row from left to right and rows sequentially from top to bottom. The contour scan selects the elements of an array so that they are adjacent to or lie on a given boundary and so that the subsequent elements are adjacent to the previously selected points. Use of the contour scanning technique, rather than the raster scanning technique, accelerates the speed of propagation of the boundary condition effects when solving partial differential equations by finite-difference methods or other iterative methods. This technique is useful in other applications, such as pattern recognition, although the implementation discussed here refers to iterative solutions of elliptic boundary value problems.

2. Basic Concept

Let R be an open rectangular two-dimensional region with a boundary C containing an open region R_1 with boundaries C_1 and C_2 as shown in Figure 1, and let the region R be discretized with S mesh points such that $S \subset R$. The algorithm described here selects all the points S_1 in a particular order such that $S_1 \subset R_0$ where $R_0 = R_1 \cup C_2$.

The selection of the points in R_0 starts from a point adjacent to a boundary point on C_1. Then all the points in R_0 adjacent to C_1 and on C_2 are selected in a sequence as they occur in the clockwise direction of a path along C_1 and on C_2. It is noted that line segments joining all the selected adjacent points form a polygon even if the boundary C_1 and C_2 do not constitute a closed contour and intersect the boundary C. Subsequently selected points in R_0 are adjacent to the previously selected points. The procedure is repeated until the last element of S_1 is found.

Special treatment is given to multiply connected boundaries.

Received 30 August 1972 and 24 June 1974.

This work was supported by the National Research Council of Canada.
Authors' address: Department of Electrical Engineering, McMaster University, Hamilton, Ont., Canada.

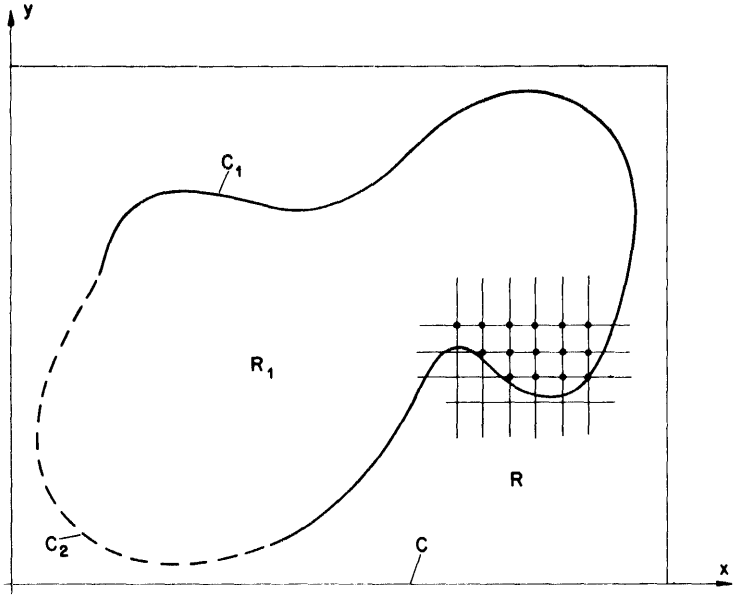


Fig. 1. Mixed nonconvex boundary value problem

3. Implementation

The simplest scanning procedure is the raster scan since it can be implemented with two DO loops. However, this procedure has two drawbacks: less favored boundary conditions affect the solution at the very end of the scan; and for nonrectangular boundaries the raster scan samples points outside the specified region where the solution is not desired.

A scanning technique with neither drawback is the contour scan shown in Figure 2. After the scan has been initialized by the proper selection of a starting point and an initial direction, it proceeds by trying to make a left turn in a two-dimensional array. If it encounters either a point on a Dirichlet boundary or a previously scanned point, it then tries to go straight ahead. If it fails, a raster scan is used to find the next available point within the boundaries. The raster scan terminates the procedure when all the points in the array have been selected.

It is seen from Figure 2 that the complexity of the path traced by the contour scan depends on the choice of the starting point. This starting point may be specified by the user. However, a more efficient way of choosing a proper starting point and the optimum contour path is guaranteed by the subroutine CONOPT. The subroutine uses two criteria for selecting proper starting points: first, it tries to choose a point adjacent to a mesh point with the highest value of Dirichlet boundary condition; second, it tries to select a point so that the contour generated is a continuous curve. This latter criterion tends to produce a consistent ordering [2].

The subroutine HIVA locates the points with large numerical values of Dirichlet boundaries by means of the raster scan. The subroutine SPOINT then selects up to twenty choices for possible starting points. The points are chosen in such a way as to minimize the number of corners in the scan and so that the scan runs adjacent, if possible, to the entire equipotential boundary. The subroutine CONTR generates different orderings from each of these points. The subroutine CONOPT chooses either the first continuous scan, or the scan with the fewest discontinuities.

The initial direction in which the contour scan proceeds from a starting point is determined automatically in the subroutine CONTR as follows. If the Dirichlet boundary point is located at $(I - 1, J)$ and the point, $(I, J + 1)$ is an interior "free" point, the initial direction indicator, K, is set to 1, and the turn indicator, L, is set to 2, i.e. the scanning proceeds in the direction of increasing J. If this condition is not met, the previous pattern is rotated about the point (I, J). Eventually, one of four possible directions will be chosen.

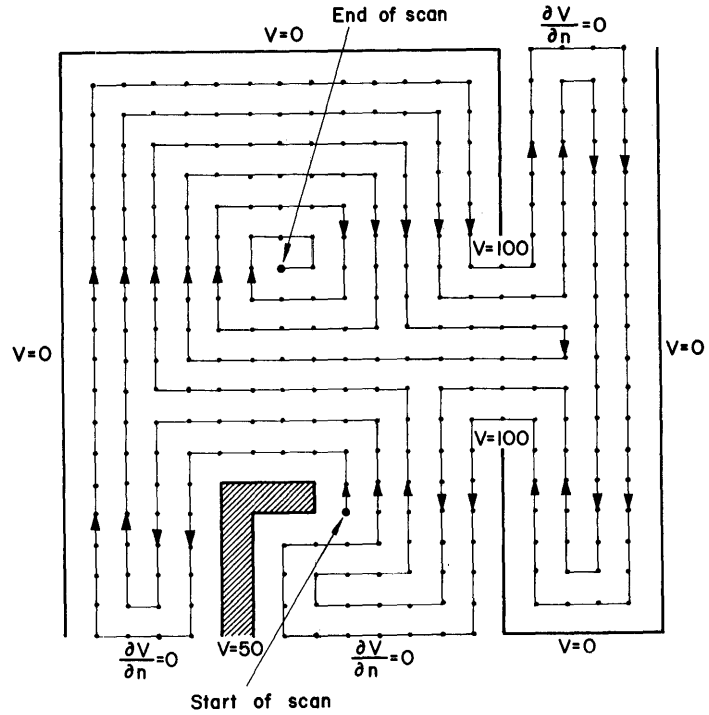


Fig. 2. Sample of boundary conditions and contour scan

The scanning path is computed only once for each different set of Dirichlet boundary values and it is stored in a tabular form. The scanning then is carried out by a single DO loop using table look-up. One can change the direction of the scan by altering the table look-up. For repeated solutions of the same problem one may punch out the selected contour scan.

4. Input-Output Considerations

In order to determine a contour scanning path, data is required in the NET array locating Dirichlet boundaries. The optimization of the contour scan requires numerical values of these boundaries. If the boundaries are piecewise linear or are described by a simple function, they may be generated by the computer. Otherwise the user has to supply them point by point. The numerical values of the boundaries are introduced in the array A and the corresponding location flags are automatically shifted in the array NET since the dimension of the array NET is greater than that of the array A. The program can handle purely Neumann problems, i.e. with the boundary consisting of C_2 only. Setting the boundary conditions in the arrays A and NET may be implemented in many different ways and it is felt that it does not belong to the described algorithm.

The intermediate results printed by the subprogram CONOPT may be suppressed by setting the flag IPRINT to zero.

5. Efficiency of the Algorithm

This algorithm has been designed to be used in iterative solutions of elliptic boundary-value problems. Therefore, its efficiency can be measured by comparing the total machine time spent for solving a given problem when using the contour scan and the raser scan. We consider two stages: selecting and storing a set of points, S_1, from the region $R_1 \cup C_2$; and solving the problem by iteration.

Since the implementation of the contour scan is more complex than the implementation of the raster scan, the first stage is more economical for the raster scan, depending on the geometry of the region R_0. Various test problems indicate that the set-up time for the contour scan is 10 to 50 times longer than that for the raster scan. However, this set-up time is only 10 to .5 times longer than the average

machine time per iteration for a small problem, as in Figure 2, and for large problems with up to 6500 points, respectively.

The average machine time per iteration during the second stage is exactly the same for both scanning techniques since the number of elements in S_1 is the same in both cases. It has been found experimentally [1] that the ordering of S_1 generating the contour scan causes a higher rate of convergence of the finite-difference methods as compared to the ordering generated by the raster scan. In all the test problems, this faster convergence resulted in 2 to 10 percent fewer iterations when using the Jacobi, Gauss-Seidel, or SOR methods. A more efficient method [1] resulted in up to 60 percent less iterations. Other scanning techniques such as the diagonal scan and the "red-black" scan [2] give the same rate of convergence as the raster scan.

The subroutine CONTR may be used as an independent subprogram for generating only one contour scanning path. In this case, it requires only 2 to 5 times more set-up time than the raster scan.

All the test problems have been run on a CDC-6400 computer.

REFERENCES

1. Della Torre, E., and Kinsner, W. A successive extrapolated relaxation (SER) method for solving partial difference equations. Presented at the Computers, Electronics and Control Conf., Calgary, Alta., Canada. May 23–25, 1974.
2. Young, D.M. *Iterative Solution of Large Linear Systems.* Academic Press, New York, 1971.

ALGORITHM

```
      SUBROUTINE CONOPT(M, N, MM, NN, MN, A, NET, IPRINT, NP, IJ)       CON   10
C THIS SUBPROGRAM OPTIMIZES THE CONTOUR SCAN AND IT RETURNS             CON   20
C THE OPTIMUM CONTOUR SCANNING PATH STORED IN THE ARRAY IJ.             CON   30
C                     .....INPUT VARIABLES.....                         CON   40
C M  -NUMBER OF SUBDIVISIONS IN X-DIRECTION. DIMENSION OF THE           CON   50
C     ARRAY A IN I DIRECTION.                                           CON   60
C N  -AS ABOVE BUT IN Y AND J DIRECTIONS.                               CON   70
C A  -WORKING ARRAY WITH NUMERICAL VALUES OF DIRICHLET                  CON   80
C     BOUNDARY CONDITIONS.  A(M,N).                                     CON   90
C MM -NUMBER OF SUBDIVISIONS IN X-DIRECTION. DIMENSION OF THE           CON  100
C     ARRAY NET IN I DIRECTION. MM=M+2.                                 CON  110
C NN -AS ABOVE BUT IN Y AND J DIRECTIONS. NN=N+2.                       CON  120
C NET-AN INTEGER ARRAY WITH FLAGS LOCATING THE DIRICHLET                CON  130
C     AND NEUMANN BOUNDARY CONDITIONS. 0=VARIABLES,                     CON  140
C     1=DIRICHLET B.C., 2=NEUMANN B.C.  NET(MM,NN). THESE               CON  150
C     FLAGS ARE SHIFTED WITH RESPECT TO THE ARRAY A SUCH                CON  160
C     THAT THE LOCATION OF A POINT ON THE DIRICHLET BOUNDARY            CON  170
C     A(I,J) CORRESPONDS TO NET(I+1,J+1).                               CON  180
C MN -DIMENSION OF THE ARRAY IJ. IT DEPENDS ON THE LENGTH               CON  190
C     OF THE CONTOUR PATH, NR. PREFERABLY MN=M*N.                       CON  200
C IPRINT -IF IPRINT=0, THE PRINTOUT IN CONOPT IS SUPPRESSED.            CON  210
C                     .....OUTPUT VARIABLES.....                        CON  220
C NP -NUMBER OF POINTS SCANNED BY THE CONTOUR PATH DETERMINED           CON  230
C     BY THE SUBROUTINE CONTR.                                          CON  240
C IJ -AN INTEGER ARRAY CONTAINING COORDINATES I,J OF THE                CON  250
C     PONTS (I,J)  SCANNED BY THE CONTOUR PATH. THEY ARE                CON  260
C     DETERMINED BY CONTR. THE ARRAY IJ HAS 3 COLUMNS, THE              CON  270
C     LAST ONE IS USED FOR COMPUTATIONAL PURPOSES IN CONTR,             CON  280
C     AND, ON THE OUTPUT, IT CONTAINS THE CORRESPONDING FLAGS           CON  290
C     FROM THE ARRAY NET.                                               CON  300
C                     .....INTERNAL VARIABLES.....                      CON  310
C K  -NUMBER OF STARTING POINTS DETERMINED BY THE SUBROUTINE            CON  320
C     SPOINT. K IS USED ONLY IN THIS SUBROUTINE.                        CON  330
C IJS-AN INTEGER ARRAY. IT CONTAINS 4 COLUMNS, THE FIRST TWO            CON  340
C     OF THEM CONTAIN COORDINATES I,J OF THE STARTING POINTS            CON  350
C     (I,J) DETERMINED BY THE SUBROUTINE SPOINT, THE THIRD              CON  360
C     ONE CONTAINS CODED VALUES OF DIRICHLET B.C. ADJACENT              CON  370
C     TO A POINT (I,J), AND THE LAST ONE CONTAINS THE NUMBER            CON  380
C     OF BREAKS IN THE CONTOUR SCAN WHICH IS INITIATED FROM             CON  390
C     THE CORRESPONDING POINT (I,J). THE NUMBER OF ROWS IN              CON  400
C     IJS CAN BE CHANGED, IF NECESSARY.                                 CON  410
C KK -NUMBER OF HIGH VALUE DIRICHLET BOUNDARY CONDITIONS,               CON  420
C     DETERMINED BY THE SUBROUTINE HIVA. KK IS USED IN SPOINT           CON  430
C AA -AN ARRAY CONTAINING THE HIGH VALUE DIRICHLET B.C.,                CON  440
C     FOUND BY HIVA. THE LENGTH OF THE ARRAY AA CAN BE                  CON  450
C     CHANGED, IF NECESSARY.                                            CON  460
C                     .....NOTES.....                                   CON  470
C THE PARAMETERS M,N,MM,NN,A,NET AND IPRINT MUST BE SUPPLIED            CON  480
C BY THE USER (INPUT). PARAMETERS NP AND IJ ARE RETURNED BY             CON  490
C THE SUBROUTINE CONOPT (OUTPUT).  THE MAIN PROGRAM COMMUNI-            CON  500
C CATES WITH THE SUBPROGRAM CONOPT THROUGH THE PARAMETER LIST           CON  510
C ONLY.  ALL THE LABELED COMMON STORAGE WITHIN THE SUBPROG-             CON  520
C RAM CONOPT IS FOR THE INTERNAL USE ONLY.                              CON  530
```

```
C IF THE CONDITIONS MM= M+2 AND NN= N+2 ARE NOT SATISFIED,          CON  540
C EXECUTION OF THE SUBPROGRAM IS INTERRUPTED AND A MESSAGE IS       CON  550
C PRINTED OUT  (AUTOMATIC ADJUSTMENT OF MM AND NN MIGHT CAUSE       CON  560
C SEVERE ERRORS IN THE MAIN PROGRAM).  IF THERE IS NO DIRICH-       CON  570
C LET BOUNDARY POINT (NO C1), THE COMPUTER AUTOMATICALLY IN-        CON  580
C SERTS SUCH A POINT IN THE REGION R1 WITHIN THE BOUNDARY C2        CON  590
C (THIS MERELY PINS DOWN ONE SOLUTION OF THE NEUMANN PROBLEM).      CON  600
C IF THERE ARE MORE INTERIOR POINTS THAN ALLOWED BY THE IN-         CON  610
C PUT PARAMETER MN, EXECUTION IS INTERRUPTED AND A MESSAGE          CON  620
C IS PRINTED OUT  (MN MAY NOT BE ADJUSTED AUTOMATICALLY BY          CON  630
C THE COMPUTER).                                                    CON  640
      DIMENSION A(M,N), NET(MM,NN), IJ(MN,3)                        CON  650
      COMMON /SPT/ K, IJS(2Ø,4) /CS/ NR /HV/ KK, AA(2Ø)             CON  660
      IF (MM.NE.M+2 .OR. NN.NE.N+2) GO TO 7Ø                        CON  670
      IP = IPRINT                                                   CON  680
      IF (IP.NE.Ø) WRITE (6,99999)                                  CON  690
C FIND PROPER STARTING POINTS FOR THE CONTOUR SCAN.                 CON  700
      CALL SPOINT(M, N, MM, NN, A, NET, IP)                         CON  710
C FIND A CONTINUOUS CONTOUR PATH, IF POSSIBLE.                      CON  720
      L = Ø                                                         CON  730
   1Ø L = L + 1                                                     CON  740
      CALL CONTR(MM, NN, MN, NET, IJ, L)                            CON  750
      NP = NR                                                       CON  760
      IF (IJS(L,4).EQ.Ø) GO TO 5Ø                                   CON  770
      IF (L.LT.K) GO TO 1Ø                                          CON  780
C SINCE THERE IS NO CONTINUOUS PATH FIND THE PATH WITH THE          CON  790
C LARGEST NUMBER OF BREAKS.                                         CON  800
      IG = Ø                                                        CON  810
      DO 2Ø L=1,K                                                   CON  820
        IF (IJS(L,4).GT.IG) IG = IJS(L,4)                           CON  830
   2Ø CONTINUE                                                      CON  840
C FIND THE PATH WITH THE SMALLEST NUMBER OF BREAKS.                 CON  850
      IS = IG                                                       CON  860
      DO 3Ø L=1,K                                                   CON  870
        IF (IJS(L,4).LT.IS) IS = IJS(L,4)                           CON  880
   3Ø CONTINUE                                                      CON  890
C CHOOSE A PATH WITH THE SMALLEST NUMBER OF BREAKS BEING            CON  900
C CLOSE TO THE HIGHEST VALUE OF DIRICHLET B.C.                      CON  910
      L = Ø                                                         CON  920
   4Ø L = L + 1                                                     CON  930
      IF (IJS(L,4).EQ.IS) GO TO 6Ø                                  CON  940
      IF (L.LT.K) GO TO 4Ø                                          CON  950
   5Ø I = IJS(L,3)                                                  CON  960
      IF (IP.NE.Ø) WRITE (6,99998) L, (IJS(L,J),J=1,2), AA(I)       CON  970
      RETURN                                                        CON  980
   6Ø I = IJS(L,3)                                                  CON  990
      IF (IP.NE.Ø) WRITE (6,99997) IS, AA(I), (IJS(L,J),J=1,2)      CON 1ØØØ
C DETERMINE THE FINAL CONTOUR PATH.                                 CON 1Ø1Ø
      CALL CONTR(MM, NN, MN, NET, IJ, L)                            CON 1Ø2Ø
      RETURN                                                        CON 1Ø3Ø
   7Ø WRITE (6,99996)                                               CON 1Ø4Ø
      STOP                                                          CON 1Ø5Ø
99999 FORMAT (1H1, 1ØX, 26HCONTOUR PATH OPTIMIZATION.//)            CON 1Ø6Ø
99998 FORMAT (/1X, 39HTHE CONTINUOUS CONTOUR SCAN HAS BEEN FO,      CON 1Ø7Ø
     * 9HUND AFTER, I3, 15H EVALUATION(S)./17H THE STARTING POI,    CON 1Ø8Ø
     * 8HNT IS A(, I2, 1H,, I2, 4H) AT, E12.4, 5H B.C.//)          CON 1Ø9Ø
99997 FORMAT (/1X, 28HTHE OPTIMUM CONTOUR SCAN HAS, I4, 2H B,       CON 11ØØ
     * 17HREAK(S) (MINIMUM)/3ØH AND IT STARTS FROM THE HIGHES,     CON 111Ø
     * 17HT POSSIBLE VALUE,, E12.4/25H THE STARTING POINT IS A(,   CON 112Ø
     * I2, 1H,, I2, 2H).//)                                        CON 113Ø
99996 FORMAT (1H1, 1X, 36HADJUST THE PARAMETERS MM AND NN SUCH,     CON 114Ø
     * 5H THAT/1X, 8HMM = M+2/1X, 8HNN = N+2//1X, 14HSTOP FROM CONO, CON 115Ø
     * 3HPT.)                                                      CON 116Ø
      END                                                           CON 117Ø

      SUBROUTINE SPOINT(M, N, MM, NN, A, NET, IP)                   SPO   1Ø
C THE SUBROUTINE DETERMINES THE POSITION OF POINTS IN THE           SPO   2Ø
C VICINITY OF THE HIGHEST AND INTERMEDIATE VALUES OF DIRICHLET      SPO   3Ø
C BOUNDARY CONDITIONS. THE ARRAYS A AND NET SERVE AS AN INPUT.      SPO   4Ø
C ML -AN INTEGER, ML=M+1                                            SPO   5Ø
C NL -AN INTEGER, NL=N+1                                            SPO   6Ø
C THE PARAMETERS ARE DEFINED IN CONOPT.                             SPO   7Ø
      DIMENSION A(M,N), NET(MM,NN)                                  SPO   8Ø
      COMMON /COEF/ ML, NL /SPT/ K, IJS(2Ø,4) /HV/ KK, AA(2Ø)       SPO   9Ø
C FIND HIGH VALUE DIRICHLET B.C.                                    SPO  1ØØ
      CALL HIVA(M, N, MM, NN, A, NET, IP)                           SPO  11Ø
      K = Ø                                                         SPO  12Ø
      DO 1ØØ L=1,KK                                                 SPO  13Ø
```

```
      AM = AA(L)                                                    SPO  14Ø
C FIND POINTS SEPARATED FROM THE HIGH VALUE DIRICHLET B.C.          SPO  15Ø
C BY ONE MESH LENGTH. THESE POINTS LIE ON THE ROWS.                 SPO  16Ø
      DO 3Ø I=1,M                                                   SPO  17Ø
        DO 2Ø J=1,N                                                 SPO  18Ø
          IF (NET(I+1,J+1).NE.Ø) GO TO 2Ø                           SPO  19Ø
          IF (A(I-1,J).EQ.AM .AND. NET(I,J).NE.1) GO TO 1Ø          SPO  2ØØ
          IF (A(I+1,J).NE.AM .OR. NET(I+2,J+2).EQ.1) GO TO 2Ø       SPO  21Ø
   1Ø     K = K + 1                                                 SPO  22Ø
          IF (K.GT.2Ø) GO TO 11Ø                                    SPO  23Ø
          IJS(K,1) = I                                              SPO  24Ø
          IJS(K,2) = J                                              SPO  25Ø
          IJS(K,3) = L                                              SPO  26Ø
   2Ø   CONTINUE                                                    SPO  27Ø
   3Ø CONTINUE                                                      SPO  28Ø
C FIND POINTS AS ABOVE BUT LYING ON THE COLUMNS.                    SPO  29Ø
      DO 6Ø J=1,N                                                   SPO  3ØØ
        DO 5Ø I=1,M                                                 SPO  31Ø
          IF (NET(I+1,J+1).NE.Ø) GO TO 5Ø                           SPO  32Ø
          IF (A(I,J+1).EQ.AM .AND. NET(I,J+2).NE.1) GO TO 4Ø        SPO  33Ø
          IF (A(I,J-1).NE.AM .OR. NET(I+2,J).EQ.1) GO TO 5Ø         SPO  34Ø
   4Ø     K = K + 1                                                 SPO  35Ø
          IF (K.GT.2Ø) GO TO 11Ø                                    SPO  36Ø
          IJS(K,1) = I                                              SPO  37Ø
          IJS(K,2) = J                                              SPO  38Ø
          IJS(K,3) = L                                              SPO  39Ø
   5Ø   CONTINUE                                                    SPO  4ØØ
   6Ø CONTINUE                                                      SPO  41Ø
C FIND POINTS ADJACENT TO DIRICHLET B.C., LOCATED AT CORNERS        SPO  42Ø
      DO 9Ø I=2,ML                                                  SPO  43Ø
        DO 8Ø J=2,NL                                                SPO  44Ø
          IF (NET(I,J).NE.Ø) GO TO 8Ø                               SPO  45Ø
          IF (A(I-2,J-1).EQ.AM .AND. NET(I-1,J-1).EQ.1 .AND.        SPO  46Ø
     *     NET(I,J-1).EQ.1) GO TO 7Ø                                SPO  47Ø
          IF (A(I-1,J).EQ.AM .AND. NET(I-1,J+1).EQ.1 .AND.          SPO  48Ø
     *     NET(I-1,J).EQ.1) GO TO 7Ø                                SPO  49Ø
          IF (A(I,J-1).EQ.AM .AND. NET(I+1,J+1).EQ.1 .AND.          SPO  5ØØ
     *     NET(I,J+1).EQ.1) GO TO 7Ø                                SPO  51Ø
          IF (A(I-1,J-2).NE.AM .OR. NET(I+1,J-1).NE.1 .OR.          SPO  52Ø
     *     NET(I+1,J).NE.1) GO TO 8Ø                                SPO  53Ø
   7Ø     K = K + 1                                                 SPO  54Ø
          IF (K.GT.2Ø) GO TO 11Ø                                    SPO  55Ø
          IJS(K,1) = I - 1                                          SPO  56Ø
          IJS(K,2) = J - 1                                          SPO  57Ø
          IJS(K,3) = L                                              SPO  58Ø
   8Ø   CONTINUE                                                    SPO  59Ø
   9Ø CONTINUE                                                      SPO  6ØØ
  1ØØ CONTINUE                                                      SPO  61Ø
  11Ø IF(IP.NE.Ø) WRITE(6,99999) (I,(IJS(I,J), J=1,2), I=1,K)       SPO  62Ø
      RETURN                                                        SPO  63Ø
99999 FORMAT (/1X, 39HSTARTING POINTS FOR CONTOUR OPTIMIZATIO, 1HN//  SPO  64Ø
     * 5X, 2HNO, 5X, 1HI, 5X, 1HJ/(I7, 2I6))                       SPO  65Ø
      END                                                           SPO  66Ø


      SUBROUTINE HIVA(M, N, MM, NN, A, NET, IP)                     HIV   1Ø
C THE SUBROUTINE DETERMINES THE HIGH VALUE DIRICHLET BOUNDA-        HIV   2Ø
C RY CONDITIONS AND IT RETURNS THEM IN THE ARRAY AA.                HIV   3Ø
C PARAMETERS ARE DEFINED IN SPOINT.                                 HIV   4Ø
      DIMENSION A(M,N), NET(MM,NN)                                  HIV   5Ø
      COMMON /HV/ KK, AA(2Ø)                                        HIV   6Ø
C FIND THE MAXIMUM AND MINIMUM VALUES IN THE ARRAY A.               HIV   7Ø
C ARE THERE ANY POINTS OF TYPE S1                                   HIV   8Ø
      KK = Ø                                                        HIV   9Ø
      DO 2Ø J=1,N                                                   HIV  1ØØ
        J1 = J + 1                                                  HIV  11Ø
        DO 1Ø I=1,M                                                 HIV  12Ø
          IF (NET(I+1,J1).EQ.1) KK = KK + 1                         HIV  13Ø
   1Ø   CONTINUE                                                    HIV  14Ø
   2Ø CONTINUE                                                      HIV  15Ø
C IF KK=Ø (NO S1 POINTS), THE SOLUTION WILL BE PINNED DOWN.         HIV  16Ø
      IF (KK) 3Ø, 3Ø, 6Ø                                           HIV  17Ø
   3Ø DO 5Ø J=1,N                                                   HIV  18Ø
```

```
      J1 = J + 1                                                          HIV  19Ø
      DO 4Ø I=1,M                                                         HIV  2ØØ
        IF (NET(I+1,J1).NE.Ø) GO TO 4Ø                                    HIV  21Ø
        NET(I+1,J1) = 1                                                   HIV  22Ø
        A(I,J) = 5Ø.Ø                                                     HIV  23Ø
        AA(1) = 5Ø.Ø                                                      HIV  24Ø
        KK = 1                                                            HIV  25Ø
        GO TO 12Ø                                                         HIV  26Ø
 4Ø   CONTINUE                                                            HIV  27Ø
 5Ø CONTINUE                                                              HIV  28Ø
    WRITE (6,99999)                                                       HIV  29Ø
    STOP                                                                  HIV  3ØØ
 6Ø AM = Ø.Ø                                                              HIV  31Ø
    AS = 1ØØ.Ø                                                            HIV  32Ø
    DO 8Ø J=1,N                                                           HIV  33Ø
      J1 = J + 1                                                          HIV  34Ø
      DO 7Ø I=1,M                                                         HIV  35Ø
        IF (NET(I+1,J1).NE.1) GO TO 7Ø                                    HIV  36Ø
        IF (A(I,J).GT.AM) AM = A(I,J)                                     HIV  37Ø
        IF (A(I,J).LT.AS) AS = A(I,J)                                     HIV  38Ø
 7Ø   CONTINUE                                                            HIV  39Ø
 8Ø CONTINUE                                                              HIV  4ØØ
    KK = 1                                                                HIV  41Ø
    AA(1) = AM                                                            HIV  42Ø
C FIND INTERMEDIATE VALUES.                                               HIV  43Ø
      DO 11Ø K=1,19                                                       HIV  44Ø
        AM = AS                                                           HIV  45Ø
        DO 1ØØ J=1,N                                                      HIV  46Ø
          J1 = J + 1                                                      HIV  47Ø
          DO 9Ø I=1,M                                                     HIV  48Ø
            IF (NET(I+1,J1).NE.1) GO TO 9Ø                                HIV  49Ø
            IF (A(I,J).GT.AM .AND. A(I,J).LT.AA(K)) AM = A(I,J)           HIV  5ØØ
 9Ø       CONTINUE                                                        HIV  51Ø
1ØØ     CONTINUE                                                          HIV  52Ø
        IF (AM.EQ.AS) GO TO 12Ø                                           HIV  53Ø
        KK = KK + 1                                                       HIV  54Ø
        AA(KK) = AM                                                       HIV  55Ø
11Ø CONTINUE                                                              HIV  56Ø
12Ø IF (IP.NE.Ø) WRITE (6,99998) (I,AA(I),I=1,KK)                         HIV  57Ø
    RETURN                                                                HIV  58Ø
99999 FORMAT (/1X, 37HTHERE ARE NO FREE POINTS WITHIN A AND,               HIV  59Ø
   * 26H NET. SOLUTION IMPOSSIBLE.//1X, 14HSTOP FROM HIVA,                HIV  6ØØ
   * 11H IN CONOPT.)                                                      HIV  61Ø
99998 FORMAT (/1X, 25HHIGH VALUE DIRICHLET B.C./(I7, E15.5))              HIV  62Ø
      END                                                                 HIV  63Ø
```

```
      SUBROUTINE CONTR(MM, NN, MN, NET, IJ, L1)                           CON   1Ø
C THE SUBROUTINE CONTR TRACES A CONTOUR SCAN WHICH COVERS                 CON   2Ø
C ONLY THE INTERIOR OF THE REGION DEFINED BY BOUNDARY OF AN               CON   3Ø
C ARBITRARY SHAPE. THE SUBROUTINE REQUIRES A STARTING POINT,              CON   4Ø
C I,J,AND IT RETURNS THE CONTOUR PATH STORED IN THE ARRAY IJ              CON   5Ø
C IX -AN ARRAY USED ONLY IN THIS SUBROUTINE FOR SETTING                   CON   6Ø
C     A DIRECTIONAL PATTERN OF I.                                         CON   7Ø
C JY -AS ABOVE, BUT FOR J.                                                CON   8Ø
C L  -AN INDICATOR OF THE PERFORMED TURN. L=1-TURN LEFT,                  CON   9Ø
C     L=2-GO STRAIGHT, L=3-TURN RIGHT.                                    CON  1ØØ
C K  -AN INDICATOR OF THE CHOSEN DIRECTION.K IS RELATED TO L              CON  11Ø
C OTHER PARAMETERS AS DEFINED IN CONOPT.                                  CON  12Ø
      DIMENSION NET(MM,NN), IJ(MN,3)                                      CON  13Ø
      COMMON /COEF/ ML, NL /TEMP/ IX(5), JY(5)                            CON  14Ø
      COMMON /SPT/ K1, IJS(2Ø,4) /CS/ NR                                  CON  15Ø
C SET INITIAL PARAMETERS.                                                 CON  16Ø
      I = IJS(L1,1) + 1                                                   CON  17Ø
      J = IJS(L1,2) + 1                                                   CON  18Ø
      L = 2                                                               CON  19Ø
      NR = Ø                                                              CON  2ØØ
      IJS(L1,4) = Ø                                                       CON  21Ø
      GO TO 3Ø                                                            CON  22Ø
C FIND INITIAL DIRECTION OF THE SCAN.                                     CON  23Ø
 1Ø IX(5) = IX(1)                                                         CON  24Ø
      JY(5) = JY(1)                                                       CON  25Ø
      DO 2Ø K=1,4                                                         CON  26Ø
        I = IX(K+1)                                                       CON  27Ø
        J = JY(K+1)                                                       CON  28Ø
```

```
      I1 = IX(K)                                                        CON  29Ø
      J1 = JY(K)                                                        CON  3ØØ
       IF (NET(I,J).NE.1 .AND. NET(I1,J1).EQ.1) GO TO 3Ø                CON  31Ø
   2Ø CONTINUE                                                          CON  32Ø
   3Ø NR = NR + 1                                                       CON  33Ø
      IF (NR.GT.MN) GO TO 11Ø                                           CON  34Ø
C SAVE I,J, AND NET(I,J). PROTECT NET(I,J).                             CON  35Ø
      IJ(NR,1) = I - 1                                                  CON  36Ø
      IJ(NR,2) = J - 1                                                  CON  37Ø
      IJ(NR,3) = NET(I,J)                                               CON  38Ø
      NET(I,J) = 1                                                      CON  39Ø
C SET THE DIRECTION PATTERN.                                            CON  4ØØ
      IX(1) = I - 1                                                     CON  41Ø
      JY(1) = J                                                         CON  42Ø
      IX(2) = I                                                         CON  43Ø
      JY(2) = J + 1                                                     CON  44Ø
      IX(3) = I + 1                                                     CON  45Ø
      JY(3) = J                                                         CON  46Ø
      IX(4) = I                                                         CON  47Ø
      JY(4) = J - 1                                                     CON  48Ø
      IF (NR.EQ.1) GO TO 1Ø                                             CON  49Ø
C COMPUTE THE RELATIVE DIRECTION.                                       CON  5ØØ
      K2 = K + L - 2                                                    CON  51Ø
      K = K2 + 4*(1-((3+K2)/4))                                         CON  52Ø
      K2 = K - 1                                                        CON  53Ø
      IF (K2.EQ.Ø) GO TO 6Ø                                             CON  54Ø
C ROTATE THE PATTERN ACCORDING TO THE RELATIVE DIRECTION.               CON  55Ø
      DO 5Ø L2=1,K2                                                     CON  56Ø
        IX(5) = IX(1)                                                   CON  57Ø
        JY(5) = JY(1)                                                   CON  58Ø
        DO 4Ø L=1,4                                                     CON  59Ø
          IX(L) = IX(L+1)                                               CON  6ØØ
          JY(L) = JY(L+1)                                               CON  61Ø
   4Ø   CONTINUE                                                        CON  62Ø
   5Ø CONTINUE                                                          CON  63Ø
C CHECK ALL THE POSSIBLE MOVES. IF A MOVE IS POSSIBLE, THE              CON  64Ø
C VALUE OF L WILL BE PRESERVED.                                         CON  65Ø
   6Ø DO 7Ø L=1,3                                                       CON  66Ø
        I = IX(L)                                                       CON  67Ø
        J = JY(L)                                                       CON  68Ø
        IF (NET(I,J).NE.1) GO TO 3Ø                                     CON  69Ø
   7Ø CONTINUE                                                          CON  7ØØ
C FIND AND COUNT POSSIBLE BREAKS IN THE SCAN.                           CON  71Ø
      K = 1                                                             CON  72Ø
      L = 2                                                             CON  73Ø
      DO 9Ø I=2,ML                                                      CON  74Ø
        DO 8Ø J=2,NL                                                    CON  75Ø
          IF (NET(I,J).NE.1) IJS(L1,4) = IJS(L1,4) + 1                  CON  76Ø
          IF (NET(I,J).NE.1) GO TO 3Ø                                   CON  77Ø
   8Ø   CONTINUE                                                        CON  78Ø
   9Ø CONTINUE                                                          CON  79Ø
C RESET THE ARRAY NET                                                   CON  8ØØ
      DO 1ØØ K=1,NR                                                     CON  81Ø
        I = IJ(K,1) + 1                                                 CON  82Ø
        J = IJ(K,2) + 1                                                 CON  83Ø
        NET(I,J) = IJ(K,3)                                              CON  84Ø
  1ØØ CONTINUE                                                          CON  85Ø
      RETURN                                                            CON  86Ø
  11Ø WRITE (6,99999)                                                   CON  87Ø
      STOP                                                              CON  88Ø
99999 FORMAT (/1X, 35HTHERE ARE MORE INTERIOR POINTS THAN, 6H ALLOW,     CON  89Ø
     * 9HED BY MN./1X, 19HINCREASE MN TO M*N./1X, 14HSTOP FROM CONT,     CON  9ØØ
     * 12HR IN CONOPT.)                                                 CON  91Ø
      END                                                               CON  92Ø



      SUBROUTINE RASTER(MM, NN, MN, NET, IJ)                            RAS   1Ø
C THE SUBROUTINE RETURNS A RASTER SCAN OF THE INTERIOR OF AN MXN        RAS   2Ø
C ARRAY. ONLY ACTIVE POINTS OF THE SCAN ARE SAVED IN IJ.                RAS   3Ø
      DIMENSION NET(MM,NN), IJ(MN,3)                                    RAS   4Ø
      COMMON /COEF/ ML, NL /CS/ NR                                      RAS   5Ø
      WRITE (6,99999)                                                   RAS   6Ø
      NR = Ø                                                            RAS   7Ø
      DO 2Ø I=2,ML                                                      RAS   8Ø
```

```
        DO 1Ø J=2,NL                                                    RAS   9Ø
          IF (NET(I,J).EQ.1) GO TO 1Ø                                   RAS  1ØØ
          NR = NR + 1                                                   RAS  11Ø
          IJ(NR,1) = I - 1                                              RAS  12Ø
          IJ(NR,2) = J - 1                                              RAS  13Ø
   1Ø   CONTINUE                                                        RAS  14Ø
   2Ø CONTINUE                                                          RAS  15Ø
      RETURN                                                            RAS  16Ø
99999 FORMAT (1H1, 1ØX, 11HRASTER SCAN)                                 RAS  17Ø
      END                                                               RAS  18Ø


      SUBROUTINE SPIRAL(M, N, MM, NN, MN, NET, IJ)                      SPI   1Ø
C THE SUBROUTINE TRACES AN INWARD SPIRAL PATH WITHIN AN                 SPI   2Ø
C M X N ARRAY A. THE SCAN IS RETURNED IN IJ.                            SPI   3Ø
C THE PARAMETERS ARE DEFINED IN THE SUBROUTINE CONOPT AND SPOINT.       SPI   4Ø
      DIMENSION NET(MM,NN), IJ(MN,3)                                    SPI   5Ø
      COMMON /COEF/ ML, NL /CS/ NR                                      SPI   6Ø
      WRITE (6,99999)                                                   SPI   7Ø
      NR = Ø                                                            SPI   8Ø
C FIND THE NUMBER OF RECTANGLES WHICH WILL CONSTITUTE                   SPI   9Ø
C THE SPIRAL SCAN.                                                      SPI  1ØØ
      IH = MINØ(ML,NL)/2                                                SPI  11Ø
C BEGIN FROM THE LARGEST RECTANGLE AND SUCCESSIVELY REDUCE              SPI  12Ø
C ITS SIZE.                                                             SPI  13Ø
      DO 9Ø I1=1,IH                                                     SPI  14Ø
        IS = I1                                                         SPI  15Ø
        K = Ø                                                           SPI  16Ø
C LOCATE START AND END POINTS FOR EACH SIDE OF A RECTANGLE.             SPI  17Ø
        DO 8Ø I2=1,2                                                    SPI  18Ø
          I = IS                                                        SPI  19Ø
          IF (I2.EQ.2) I = IE + 1                                       SPI  2ØØ
          IE = N - IS                                                   SPI  21Ø
          DO 7Ø I3=1,2                                                  SPI  22Ø
            K = K + 1                                                   SPI  23Ø
C SCAN ONE SIDE OF A RECTANLE AND DISTRIBUTE THE SEQUENCE OF            SPI  24Ø
C POINTS TO I OR TO J.                                                  SPI  25Ø
            DO 6Ø I4=IS,IE                                              SPI  26Ø
              GO TO (1Ø, 2Ø, 3Ø, 4Ø), K                                 SPI  27Ø
   1Ø         J = I4                                                    SPI  28Ø
              GO TO 5Ø                                                  SPI  29Ø
   2Ø         I = I4                                                    SPI  3ØØ
              GO TO 5Ø                                                  SPI  31Ø
   3Ø         J = NL - I4                                               SPI  32Ø
              GO TO 5Ø                                                  SPI  33Ø
   4Ø         I = ML - I4                                               SPI  34Ø
   5Ø         IF (NET(I+1,J+1).EQ.1) GO TO 6Ø                           SPI  35Ø
              NR = NR + 1                                               SPI  36Ø
              IJ(NR,1) = I                                              SPI  37Ø
              IJ(NR,2) = J                                              SPI  38Ø
   6Ø       CONTINUE                                                    SPI  39Ø
            J = IE + 1                                                  SPI  4ØØ
            IF (I2.EQ.2) J = IS                                         SPI  41Ø
            IE = M - IS                                                 SPI  42Ø
   7Ø     CONTINUE                                                     SPI  43Ø
   8Ø   CONTINUE                                                       SPI  44Ø
   9Ø CONTINUE                                                         SPI  45Ø
      RETURN                                                            SPI  46Ø
99999 FORMAT (1H1, 1ØX, 11HSPIRAL SCAN)                                 SPI  47Ø
      END                                                               SPI  48Ø


      SUBROUTINE SORCAR(M, N, MN, A, IJ, C, IP, ER)                     SOR   1Ø
C THIS SUBROUTINE OPTIMIZES THE OVERRELAXATION FACTOR USING CARRE*S     SOR   2Ø
C TECHNIQUE.                                                            SOR   3Ø
      DIMENSION A(M,N), IJ(MN,3), C(MN), GX(3)                          SOR   4Ø
      COMMON /LOG/ Q, QQ                                                SOR   5Ø
      LOGICAL Q, QQ, QQQ                                                SOR   6Ø
      WRITE (6,99999)                                                   SOR   7Ø
C THE FIRST ITERATION USES BETA=1.Ø                                      SOR   8Ø
      BETA = 1.                                                         SOR   9Ø
C THE FIRST INTERVAL OF INT ITERATIONS USES BETA=1.375                   SOR  1ØØ
      BOPT = 1.375                                                      SOR  11Ø
C SET THE INTERVAL OF ITERATIONS                                        SOR  12Ø
```

```
      INT = 12                                                          SOR  13Ø
      Q = .TRUE.                                                        SOR  14Ø
      QQQ = .FALSE.                                                     SOR  15Ø
      IN = Ø                                                            SOR  16Ø
      SUM = Ø.Ø                                                         SOR  17Ø
      JOB = Ø                                                           SOR  18Ø
   1Ø JOB = JOB + 1                                                     SOR  19Ø
      IN = IN + 1                                                       SOR  2ØØ
      QQ = IN.LT.INT                                                    SOR  21Ø
      BIG = Ø.Ø                                                         SOR  22Ø
      S = SUM                                                           SOR  23Ø
      SUM = Ø.Ø                                                         SOR  24Ø
      CALL SOR(M, N, MN, A, IJ, C, BETA, SUM, BIG)                      SOR  25Ø
      QQ = (.NOT.QQ) .AND. QQQ                                          SOR  26Ø
C Q=.FALSE. IF BETA OPTIMUM IS FOUND.                                   SOR  27Ø
      IF (.NOT.Q) GO TO 6Ø                                              SOR  28Ø
C QQ=.TRUE. IF JOB.GT.INT+1 AND IN=INT                                  SOR  29Ø
      IF (QQ) GO TO 3Ø                                                  SOR  3ØØ
C QQQ=.TRUE. IF JOB.GT.INT+1                                            SOR  31Ø
      IF (QQQ) GO TO 1Ø                                                 SOR  32Ø
      IF (JOB.EQ.1) BETA = BOPT                                         SOR  33Ø
      IF (JOB.EQ.1) IN = Ø                                              SOR  34Ø
      IF (IN.LE.INT-3) GO TO 1Ø                                         SOR  35Ø
C COMPUTE THREE SUCCESSIVE RATIOS OF THE DISPLACEMENT VECTOR NORMS       SOR  36Ø
C AT THE END OF THE FIRST INTERVAL                                      SOR  37Ø
      L = IN - (INT-3)                                                  SOR  38Ø
      GX(L) = SUM/S                                                     SOR  39Ø
      IF (IN.LT.INT) GO TO 1Ø                                           SOR  4ØØ
      QQQ = .TRUE.                                                      SOR  41Ø
      D1 = GX(1) - GX(2)                                                SOR  42Ø
      D2 = GX(2) - GX(3)                                                SOR  43Ø
C CRITERION FOR DETERMINING THE FEASIBILITY OF AITKEN*S                 SOR  44Ø
C EXTRAPOLATION, THE DIFFERENCES BETWEEN THE RATIOS MUST DECREASE       SOR  45Ø
C AND HAVE THE SAME SIG                                                 SOR  46Ø
      IF (ABS(D1).LE.ABS(D2) .OR. (D1/D2).LT.Ø.Ø) GO TO 2Ø              SOR  47Ø
      GAM = GX(1) - ((GX(2)-GX(1))**2/(GX(1)-2.*GX(2)+GX(3)))           SOR  48Ø
      GO TO 4Ø                                                          SOR  49Ø
   2Ø WRITE (6,99998)                                                   SOR  5ØØ
   3Ø GAM = SUM/S                                                       SOR  51Ø
   4Ø IN = Ø                                                            SOR  52Ø
      BL = BOPT                                                         SOR  53Ø
C COMPUTE A NEW ESTIMATE OF BETA OPTIMUM                                SOR  54Ø
      BOPT = 2./(1.+SQRT(ABS(1.-(GAM+BETA-1.)**2/(GAM*BETA**2))))       SOR  55Ø
C THE FOLLOWING BOPT REDUCTION PREVENTS OVERESTIMATION OF BETA          SOR  56Ø
      BETA = BOPT - (2.-BOPT)/4.                                        SOR  57Ø
C CRITERION FOR STOPPING THE PROCESS OF IMPROVING BETA.                 SOR  58Ø
      IF (ABS(BOPT-BL)/(2.-BOPT).GT.Ø.Ø5) GO TO 5Ø                      SOR  59Ø
C BETA OPTIMUM HAS BEEN FOUND                                           SOR  6ØØ
      Q = .FALSE.                                                       SOR  61Ø
      IN = INT                                                          SOR  62Ø
      BETA = BOPT                                                       SOR  63Ø
      GAM = BETA - 1.                                                   SOR  64Ø
      GR = GAM/(2.-BETA)                                                SOR  65Ø
      WRITE (6,99997) BETA, JOB                                         SOR  66Ø
   5Ø IF (GAM.EQ.1.) GO TO 1Ø                                           SOR  67Ø
      ERM = BIG*GAM/(1.-GAM)                                            SOR  68Ø
      GO TO 7Ø                                                          SOR  69Ø
   6Ø ERM = BIG*GR                                                      SOR  7ØØ
   7Ø IF (ABS(ERM).LE.ER) GO TO 8Ø                                      SOR  71Ø
      IF (JOB.LT.IP) GO TO 1Ø                                           SOR  72Ø
   8Ø WRITE (6,99996) JOB, ERM                                          SOR  73Ø
C IF BETA OPTIMUM HAS NOT BEEN FOUND YET, BUT ERM.LE.ER, PRINT BETA.    SOR  74Ø
      IF (Q) WRITE (6,99995) BETA                                       SOR  75Ø
      RETURN                                                            SOR  76Ø
99999 FORMAT (//1X, 3ØHSOLUTION BY SOR (CARRE) METHOD)                  SOR  77Ø
99998 FORMAT (/5X, 45HAITKEN*S EXTRAPOLATION NOT FEASIBLE FOR BETA ,    SOR  78Ø
     * 13HOPTIMIZATION.)                                               SOR  79Ø
99997 FORMAT (5X, 14HBETA OPTIMUM =, E14.7, 6H AFTER, I4, 7H ITERAT,    SOR  8ØØ
     * 5HIONS.)                                                        SOR  81Ø
99996 FORMAT (/1ØX, 22HNUMBER OF ITERATIONS =, I4/1ØX, 9HMAXIMUM E,     SOR  82Ø
     * 4HRROR, 8X, 1H=, E1Ø.3, 8H PERCENT)                             SOR  83Ø
99995 FORMAT (1ØX, 22HBETA (NOT OPTIMUM)   =, E14.7)                   SOR  84Ø
      END                                                               SOR  85Ø
```

```
      SUBROUTINE SOR(M, N, MN, A, IJ, C, BETA, SUM, BIG)                SOR  1Ø
C THIS IS THE SUCCESSIVE OVERRELAXATION ALGORITHM.                      SOR  2Ø
C THE SUBROUTINE PERFORMS ONE COMPLETE SWEEP OF THE INTERIOR, AND       SOR  3Ø
C COMPUTES THE ERROR CRITERION COMPONENTS.                              SOR  4Ø
      DIMENSION A(M,N), IJ(MN,3), C(MN)                                 SOR  5Ø
      COMMON /CS/ NR /LOG/ Q, QQ                                        SOR  6Ø
      EXTERNAL AV                                                       SOR  7Ø
      LOGICAL Q, QQ                                                     SOR  8Ø
      DO 1Ø K=1,NR                                                      SOR  9Ø
        I = IJ(K,1)                                                     SOR 1ØØ
        J = IJ(K,2)                                                     SOR 11Ø
        L = IJ(K,3)                                                     SOR 12Ø
        AL = A(I,J)                                                     SOR 13Ø
        AN = AV(M,N,MN,A,C,I,J,K,L)                                     SOR 14Ø
        AN = AL + BETA*(AN-AL)                                          SOR 15Ø
        A(I,J) = AN                                                     SOR 16Ø
        RES = ABS(AN-AL)                                                SOR 17Ø
        IF (Q) SUM = SUM + RES                                          SOR 18Ø
        IF (QQ) GO TO 1Ø                                                SOR 19Ø
        IF (RES.GT.BIG) BIG = RES                                       SOR 2ØØ
   1Ø CONTINUE                                                          SOR 21Ø
      RETURN                                                            SOR 22Ø
      END                                                               SOR 23Ø
```

```
      FUNCTION AV(M, N, MN, A, C, I, J, K, L)                           AV   1Ø
      DIMENSION A(M,N), C(MN)                                           AV   2Ø
      GO TO (1Ø, 2Ø, 3Ø, 4Ø, 5Ø), L                                     AV   3Ø
C NEUMANN B.C. ARE PARALLEL TO J-AXIS                                   AV   4Ø
   1Ø AV = (2.*A(I+1,J)+C(K)+C(K)+A(I,J+1)+A(I,J-1))/4.                 AV   5Ø
      RETURN                                                            AV   6Ø
C NEUMANN B.C. ARE PARALLEL TO I-AXIS                                   AV   7Ø
   2Ø AV = (2.*A(I,J+1)+C(K)+C(K)+A(I+1,J)+A(I-1,J))/4.                 AV   8Ø
      RETURN                                                            AV   9Ø
C NEUMANN B.C. AS IN 1, BUT FOR I.GT.IN 1                               AV  1ØØ
   3Ø AV = (2.*A(I-1,J)+C(K)+C(K)+A(I,J+1)+A(I,J-1))/4.                 AV  11Ø
      RETURN                                                            AV  12Ø
C NEUMANN B.C. AS IN 2, BUT FOR J.GT. IN 2                              AV  13Ø
   4Ø AV = (2.*A(I,J-1)+C(K)+C(K)+A(I+1,J)+A(I-1,J))/4.                 AV  14Ø
      RETURN                                                            AV  15Ø
C ORDINARY 5-POINT OPERATOR FOR THE INTERIOR.                           AV  16Ø
   5Ø AV = (A(I,J+1)+A(I+1,J)+A(I-1,J)+A(I,J-1))/4.                     AV  17Ø
      RETURN                                                            AV  18Ø
      END                                                               AV  19Ø
```

```
      SUBROUTINE BOUND(M, N, MM, NN, A, NET)                            BOU  1Ø
C THE SUBROUTINE READS THE BOUNDARY CONDITIONS SPECIFIED IN THE         BOU  2Ø
C DATA CARDS AND SETS THEM INTO THE ARRAYS A AND NET. THE BOUNDARY      BOU  3Ø
C CONDITIONS ARE DETERMINED BY SEGMENTS P1P2 WHERE POINTS P1(I1,J1)     BOU  4Ø
C AND P2(I2,J2) COINCIDE WITH THE MESH POINTS IMPOSED ON THE REGION     BOU  5Ø
C A(M,N). THE POINTS MUST SATISFY THE FOLLOWING CONDITION P1.LE.P2.     BOU  6Ø
C THE VALUES AB ASSOCIATED WITH THE POINTS P1 AND P2 CAN BE AB1.LE.AB2  BOU  7Ø
C OR AB1.GE.AB2. DIRICHLET B.C. ARE DISTINGUISHED FROM NEWMANN B.C.     BOU  8Ø
C BY THE INDICATOR IND WHERE IND=1 FOR DIRICHLET OR IND=2 FOR NEUMANN   BOU  9Ø
C B.C.  EACH DATA CARD CONTAINS ONLY ONE SEGMENT P1P2 (OR A POINT).     BOU 1ØØ
C THE INDEX NEXT.NE.Ø TERMINATES THE READING PROCESS FOR ONE PROBLEM.   BOU 11Ø
      DIMENSION A(M,N), NET(MM,NN)                                      BOU 12Ø
      COMMON /COEF/ ML, NL /TEMP/ ID(5), IY(5)                          BOU 13Ø
C CLEAR THE ARRAYS.                                                     BOU 14Ø
      DO 2Ø J=1,N                                                       BOU 15Ø
        DO 1Ø I=1,M                                                     BOU 16Ø
          A(I,J) = Ø.Ø                                                  BOU 17Ø
          NET(I+1,J+1) = Ø                                              BOU 18Ø
   1Ø   CONTINUE                                                        BOU 19Ø
   2Ø CONTINUE                                                          BOU 2ØØ
C SET BORDERS IN THE ARRAYS A AND NET.                                  BOU 21Ø
      DO 3Ø I=2,ML                                                      BOU 22Ø
        NET(I,2) = 1                                                    BOU 23Ø
        NET(I,NL) = 1                                                   BOU 24Ø
   3Ø CONTINUE                                                          BOU 25Ø
      DO 4Ø J=2,NL                                                      BOU 26Ø
        NET(2,J) = 1                                                    BOU 27Ø
```

```
      NET(ML,J) = 1                                                    BOU  28Ø
   4Ø CONTINUE                                                         BOU  29Ø
      DO 5Ø I=1,MM                                                     BOU  3ØØ
        NET(I,1) = 1                                                   BOU  31Ø
        NET(I,NN) = 1                                                  BOU  32Ø
   5Ø CONTINUE                                                         BOU  33Ø
      DO 6Ø J=1,NN                                                     BOU  34Ø
        NET(1,J) = 1                                                   BOU  35Ø
        NET(MM,J) = 1                                                  BOU  36Ø
   6Ø CONTINUE                                                         BOU  37Ø
   7Ø READ (5,99999) I1, J1, AB1, I2, J2, AB2, IND, NEXT               BOU  38Ø
      IF (I1.GT.I2 .OR. J1.GT.J2) GO TO 15Ø                            BOU  39Ø
C PLACE BOUNDARY PARALLEL TO I-AXIS INTO A.                            BOU  4ØØ
      IF (I1.EQ.I2) GO TO 9Ø                                           BOU  41Ø
      AB = (AB2-AB1)/FLOAT(I2-I1)                                      BOU  42Ø
      DO 8Ø I=I1,I2                                                    BOU  43Ø
        A(I,J1) = AB1 + AB*FLOAT(I-I1)                                 BOU  44Ø
C THE FOLLOWING STATEMENT TRUNCATES ABOVE 5TH DECIMAL DIGIT.           BOU  45Ø
        IF (AB.NE.Ø.Ø) A(I,J1) = AINT(A(I,J1)*1.E5)*1.E-5              BOU  46Ø
        NET(I+1,J1+1) = IND                                            BOU  47Ø
   8Ø CONTINUE                                                         BOU  48Ø
      GO TO 12Ø                                                        BOU  49Ø
C PLACE BOUNDARY PARALLEL TO J-AXIS INTO A.                            BOU  5ØØ
   9Ø IF (J1.EQ.J2) GO TO 11Ø                                          BOU  51Ø
      AB = (AB2-AB1)/FLOAT(J2-J1)                                      BOU  52Ø
      DO 1ØØ J=J1,J2                                                   BOU  53Ø
        A(I1,J) = AB1 + AB*FLOAT(J-J1)                                 BOU  54Ø
        IF (AB.NE.Ø.Ø) A(I1,J) = AINT(A(I1,J)*1.E5)*1.E-5              BOU  55Ø
        NET(I1+1,J+1) = IND                                            BOU  56Ø
  1ØØ CONTINUE                                                         BOU  57Ø
      GO TO 12Ø                                                        BOU  58Ø
C PLACE THE POINT (I1,J1) INTO A.                                      BOU  59Ø
  11Ø A(I1,J1) = AB1                                                   BOU  6ØØ
      NET(I1+1,J1+1) = IND                                             BOU  61Ø
  12Ø IF (NEXT.EQ.Ø) GO TO 7Ø                                          BOU  62Ø
C PRINT THE ARRAY NET.                                                 BOU  63Ø
      WRITE (6,99998)                                                  BOU  64Ø
      ND = 1 + (N-1)/1Ø                                                BOU  65Ø
      DO 13Ø J=1,ND                                                    BOU  66Ø
        ID(J) = J                                                      BOU  67Ø
  13Ø CONTINUE                                                         BOU  68Ø
      WRITE (6,99997) (ID(J),J=1,ND)                                   BOU  69Ø
      DO 14Ø I=1,M                                                     BOU  7ØØ
        WRITE (6,99996) I, (NET(I+1,J),J=2,NL)                         BOU  71Ø
  14Ø CONTINUE                                                         BOU  72Ø
      RETURN                                                           BOU  73Ø
  15Ø WRITE (6,99995)                                                  BOU  74Ø
      STOP                                                             BOU  75Ø
99999 FORMAT (2(2I5, F1Ø.4), 2I5)                                      BOU  76Ø
99998 FORMAT (1H1, 1ØX, 4ØHTHE FOLLOWING MATRIX REPRESENTS THE NET,,   BOU  77Ø
     * 5X, 11HØ=VARIABLES/56X, 26H1=DIRICHLET BOUNDARY COND./56X,      BOU  78Ø
     * 24H2=NEUMANN BOUNDARY COND./)                                   BOU  79Ø
99997 FORMAT (1H , 35X, 5(I1, 19X)/)                                   BOU  8ØØ
99996 FORMAT (11X, I2, 4X, 5ØI2)                                       BOU  81Ø
99995 FORMAT (1H1, 1X, 41HINCORRECT INPUT DATA, I1.GT.I2 OR J1.GT.J,   BOU  82Ø
     * 2H2.)                                                          BOU  83Ø
      END                                                              BOU  84Ø
```

```
      SUBROUTINE DINE(M, N, MM, NN, MN, A, NET, IJ, C)                 DIN   1Ø
C THE SUBROUTINE RETURNS REFERENCE MARKS LOCATED IN THE ARRAY IJ(I,3).  DIN   2Ø
C THE MARKS WILL ACCOMPANY THE SCAN AND HENCE SIMPLIFY THE LOGIC OF     DIN   3Ø
C THE ITERATION PROCESS. BOUNDARY CONDITIONS IN THE ARRAY NET ARE       DIN   4Ø
C IDENTIFIED BY 1= DIRICHLET, 2= NEUMANN, AND Ø= INTERIOR OF THE       DIN   5Ø
C REGION.                                                              DIN   6Ø
      DIMENSION A(M,N), NET(MM,NN), IJ(MN,3), C(MN)                    DIN   7Ø
      COMMON /CS/ NR                                                   DIN   8Ø
C SET MARKS OF REFERENCE, AND SAVE NEUMANN B.C. FROM A INTO C.         DIN   9Ø
C 1-4 = NEUMANN B.C. LOCATED AT (I-1,J),(I,J-1),(I+1,J), AND (I,J+1)   DIN  1ØØ
C WITH RESPECT TO THE INTERIOR OF THE REGION, RESPECTIVELY.            DIN  11Ø
C 5-INTERNAL (REGULAR) POINTS.  NOTE THAT THE MARK CHANGE FROM Ø TO 5  DIN  12Ø
C IS INTERNAL TO THE PROGRAM AND IS INTRODUCED TO SIMPLIFY THE         DIN  13Ø
C SELECTION OF EQUATIONS PERTINENT TO EITHER THE NEUMANN POINTS (1-4)  DIN  14Ø
C OR THE REGULAR POINTS (5) /SEE FUNCTION AV/.                         DIN  15Ø
```

```
      DO 3Ø K=1,NR                                                    DIN  16Ø
        I = IJ(K,1) + 1                                               DIN  17Ø
        J = IJ(K,2) + 1                                               DIN  18Ø
        IF (NET(I,J)-1) 1Ø, 4Ø, 2Ø                                    DIN  19Ø
   1Ø   IJ(K,3) = 5                                                   DIN  2ØØ
        C(K) = Ø.Ø                                                    DIN  21Ø
        GO TO 3Ø                                                      DIN  22Ø
   2Ø   IF (NET(I+1,J).EQ.Ø) IJ(K,3) = 1                             DIN  23Ø
        IF (NET(I,J+1).EQ.Ø) IJ(K,3) = 2                              DIN  24Ø
        IF (NET(I-1,J).EQ.Ø) IJ(K,3) = 3                              DIN  25Ø
        IF (NET(I,J-1).EQ.Ø) IJ(K,3) = 4                              DIN  26Ø
        C(K) = A(I-1,J-1)                                             DIN  27Ø
   3Ø CONTINUE                                                        DIN  28Ø
      WRITE (6,99999) NR                                              DIN  29Ø
      RETURN                                                          DIN  3ØØ
   4Ø WRITE (6,99998) I, J                                            DIN  31Ø
      STOP                                                            DIN  32Ø
99999 FORMAT (/1ØX, 3ØHNUMBER OF SCANNED POINTS, NR= , I5)            DIN  33Ø
99998 FORMAT (/1X, 36HINCORRECT SCANNING SEQUENCE AT NET (, I2, 1H,,  DIN  34Ø
     * I2, 2H).)                                                      DIN  35Ø
      END                                                             DIN  36Ø


      SUBROUTINE INVA(M, N, MM, NN, A, NET)                           INV   1Ø
C THE SUBROUTINE READS A CONSTANT INITIAL VALUE, AND INTRODUCES IT    INV   2Ø
C INTO THE ARRAY A. THE INITIAL VALUE DATUM FOLLOWS THE BOUNDARY      INV   3Ø
C CONDITION DATA.                                                     INV   4Ø
      DIMENSION A(M,N), NET(MM,NN)                                    INV   5Ø
      READ (5,99999) V                                                INV   6Ø
      DO 2Ø J=1,N                                                     INV   7Ø
        DO 1Ø I=1,M                                                   INV   8Ø
          IF (NET(I+1,J+1).NE.1) A(I,J) = V                           INV   9Ø
   1Ø   CONTINUE                                                      INV  1ØØ
   2Ø CONTINUE                                                        INV  11Ø
      WRITE (6,99998) V                                               INV  12Ø
      RETURN                                                          INV  13Ø
99999 FORMAT (F1Ø.Ø)                                                  INV  14Ø
99998 FORMAT (1ØX, 29HCONSTANT INITIAL VALUE,    V=, F7.2)            INV  15Ø
      END                                                             INV  16Ø


      SUBROUTINE PRES(M, N, A)                                        PRE   1Ø
C THE SUBROUTINE PRINTS NUMERICAL RESULTS FROM THE ARRAY A.           PRE   2Ø
C EACH PRINTED PAGE WILL CONTAIN 1Ø COLUMNS OF A.                     PRE   3Ø
      DIMENSION A(M,N)                                                PRE   4Ø
      COMMON /TEMP/ LA(1Ø)                                            PRE   5Ø
      DO 3Ø L=1,N,1Ø                                                  PRE   6Ø
        JS = L - 1                                                    PRE   7Ø
        JE = L + 9                                                    PRE   8Ø
        DO 1Ø I=1,1Ø                                                  PRE   9Ø
          LA(I) = JS + I                                              PRE  1ØØ
   1Ø   CONTINUE                                                      PRE  11Ø
        WRITE (6,99999) LA                                            PRE  12Ø
        DO 2Ø I=1,M                                                   PRE  13Ø
          WRITE (6,99998) I, (A(I,J),J=L,JE)                          PRE  14Ø
   2Ø   CONTINUE                                                      PRE  15Ø
   3Ø CONTINUE                                                        PRE  16Ø
      RETURN                                                          PRE  17Ø
99999 FORMAT (1H1, 15X, 1Ø(I3, 9X)/)                                  PRE  18Ø
99998 FORMAT (1H , 5H   A(, I2, 1H,, 3X, 1Ø(F1Ø.5, 2X))               PRE  19Ø
      END                                                             PRE  2ØØ


      SUBROUTINE MAP(M, N, MM, NN, A, NET)                            MAP   1Ø
C THE SUBROUTINE PRINTS THE EQUIPOTENTIAL MAP.                        MAP   2Ø
      DIMENSION A(M,N), NET(MM,NN), CA(1Ø), SP(1Ø)                    MAP   3Ø
      COMMON /TEMP/ IZ(1Ø)                                            MAP   4Ø
      DATA CA(1), CA(2), CA(3), CA(4), CA(5), CA(6), CA(7), CA(8),    MAP   5Ø
     * CA(9), CA(1Ø) /1HØ,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/,      MAP   6Ø
```

```
     * ZERO, VAL, BLANK /1HZ,1H$,1H /                                    MAP   7Ø
C READ EQUIPOTENTIALS TO BE PRINTED OUT.                                  MAP   8Ø
      READ (5,99999) (SP(I),I=1,1Ø), TOL                                  MAP   9Ø
C PREPARE THE MAP.                                                        MAP  1ØØ
      DO 5Ø J=1,N                                                         MAP  11Ø
        DO 4Ø I=1,M                                                       MAP  12Ø
          IF (NET(I+1,J+1).NE.1) GO TO 2Ø                                 MAP  13Ø
          IF (A(I,J).EQ.Ø.Ø) GO TO 1Ø                                     MAP  14Ø
          A(I,J) = VAL                                                    MAP  15Ø
          GO TO 4Ø                                                        MAP  16Ø
   1Ø     A(I,J) = ZERO                                                   MAP  17Ø
          GO TO 4Ø                                                        MAP  18Ø
   2Ø     DO 3Ø K=1,1Ø                                                    MAP  19Ø
            IF (A(I,J).LT.SP(K)-TOL .OR. A(I,J).GT.SP(K)+TOL) GO TO       MAP  2ØØ
     *       3Ø                                                           MAP  21Ø
            A(I,J) = CA(K)                                                MAP  22Ø
            GO TO 4Ø                                                      MAP  23Ø
   3Ø     CONTINUE                                                        MAP  24Ø
          A(I,J) = BLANK                                                  MAP  25Ø
   4Ø   CONTINUE                                                          MAP  26Ø
   5Ø CONTINUE                                                            MAP  27Ø
C WRITE THE MAP.                                                          MAP  28Ø
      WRITE (6,99998)                                                     MAP  29Ø
      DO 6Ø I=1,M                                                         MAP  3ØØ
        WRITE (6,99997) (A(I,J),J=1,N)                                    MAP  31Ø
   6Ø CONTINUE                                                            MAP  32Ø
C WRITE THE LIMITS FOR THE EQUIPOTENTIAL STRIPS IN THE PRINTED MAP.       MAP  33Ø
      WRITE (6,99996)                                                     MAP  34Ø
      DO 7Ø I=1,1Ø                                                        MAP  35Ø
        IZ(I) = I - 1                                                     MAP  36Ø
   7Ø CONTINUE                                                            MAP  37Ø
      DO 8Ø I=1,5                                                         MAP  38Ø
        P1 = SP(I) - TOL                                                  MAP  39Ø
        P2 = SP(I) + TOL                                                  MAP  4ØØ
        P3 = SP(I+5) - TOL                                                MAP  41Ø
        P4 = SP(I+5) + TOL                                                MAP  42Ø
        IF (P1.LT.Ø.) P1 = Ø.                                             MAP  43Ø
        WRITE (6,99995) IZ(I), P1, P2, IZ(I+5), P3, P4                    MAP  44Ø
   8Ø CONTINUE                                                            MAP  45Ø
      WRITE (6,99994)                                                     MAP  46Ø
      RETURN                                                              MAP  47Ø
C IF THE ARRAY A IS LARGER THAN 3ØX3Ø, CHANGE THE FORMAT 99997            MAP  48Ø
C TO (15X,5Ø(A1,1X)).                                                     MAP  49Ø
99999 FORMAT (1ØF7.3, F1Ø.3)                                              MAP  5ØØ
99998 FORMAT (1H1, 7X, 41HTHE FOLLOWING IS A CONTOUR MAP OF THE EQU,      MAP  51Ø
     * 31HIPOTENTIALS WITHIN GIVEN LIMITS/)                               MAP  52Ø
99997 FORMAT (15X, 2Ø(A1, 2X)/)                                           MAP  53Ø
99996 FORMAT (/13X, 8HCONTOURS)                                           MAP  54Ø
99995 FORMAT (13X, I1, 1H=, E11.4, 4H TO , E11.4, 6H  **  , I1, 1H=,      MAP  55Ø
     * E11.4, 4H TO , E11.4)                                              MAP  56Ø
99994 FORMAT (13X, 1ØHBOUNDARIES/13X, 22HZ= ZERO DIRICHLET B.C.,          MAP  57Ø
     * 8X, 4H**  , 25H$= NONZERO DIRICHLET B.C.)                          MAP  58Ø
      END                                                                 MAP  59Ø
```

500-P 1- 0

ALGORITHM 500
Minimization of Unconstrained Multivariate Functions [E4]

D. F. SHANNO and K. H. PHUA
University of Toronto, Canada

Key Words and Phrases: minimization, optimization
CR Categories: 5.15, 5.19
Language: Fortran

DESCRIPTION

Purpose

This subroutine finds a local minimum of a nonlinear function of n variables $f(x)$ where $x = (x_1, x_2, \ldots, x_n)$, $n > 1$, can be any real numbers.

Methods

Quasi-Newton methods are iterative methods which approximate Newton's method without calculating second derivatives. The iterative sequence is defined by

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} H^{(k)} g^{(k)},$$

where $g^{(k)} = \nabla f(x^{(k)})$, the gradient of f at $x^{(k)}$, $H^{(k)}$ is a matrix which is designed to approximate the inverse Hessian matrix of f at $x^{(k)}$, and $\alpha^{(k)}$ is an appropriately chosen scalar. The sequence of matrices $H^{(k)}$ is chosen to satisfy the quasi-Newton equation $H^{(k+1)} y^{(k)} = \sigma^{(k)}$, where $\sigma^{(k)} = x^{(k+1)} - x^{(k)}$ and $y^{(k)} = g^{(k+1)} - g^{(k)}$. In general, $H^{(k+1)}$ is generated by $H^{(k+1)} = H^{(k)} + D^{(k)}$, where $D^{(k)}$ is chosen to satisfy the equation $D^{(k)} y^{(k)} = \sigma^{(k)} - H^{(k)} y^{(k)}$.

The choice of $\alpha^{(k)}$ can be accomplished either by a linear-search or a step-length method. Perhaps the most complete current reference to the general theory is Powell [8]. Two different methods for determining $\alpha^{(k)}$ are incorporated and are outlined below. The Broyden-Fletcher-Shanno (BFS) matrix update (developed by Broyden [1], Fletcher [3], Goldfarb [5], Greenstadt [6], and Shanno [9]) is used to estimate $H^{(*)}$ in each of the algorithms. Dropping the superscripts, and denoting the current point by x and the subsequent point by x^*, with the corresponding notation g and g^*, and H and H^*, the BFS formula is

$$H^* = (I - \sigma y'/\sigma' y) H (I - \sigma y'/\sigma' y) + \sigma\sigma'/\sigma' y,$$

Received 31 May 1974, 8 September 1974, and 14 March 1975.

This work was supported in part by the National Research Council of Canada under Grant NRC A7910.
Authors' present addresses: D.F. Shanno, Computer Science, School of Engineering, University of Mississippi, University, MISS 38677; K.H. Phua, Department of Computer Science, Nanyang University, Upper Jurong Rd., Singapore 22.

where $\sigma = x^* - x$, $y = g^* - g$. This form of the update formula is that suggested by Fletcher [3].

Method 1. This method incorporates an inexact linear search for $\alpha^{(k)}$ at each step. Features of the algorithm include the following.

(i) The initial approximate inverse Hessian matrix proposed by Powell [7] is used, that is, $H^{(0)} = 100 \, ||\Delta x|| / ||g^{(0)}|| \, I$, where Δx is the vector of step bounds placed on all variables x_i, $||\cdot||$ is the usual L_2-norm, and I is the $n \times n$ identity matrix.

(ii) In Shanno et al. [11], it has been shown that if α is chosen to satisfy $\sigma' g^* > \sigma' g$, then the positive definiteness of H implies that H^* is also positive definite. Thus α is chosen by the following algorithm. The quadratic extrapolation technique described in [10] is used to bracket the minimum of $f(x)$ along the search direction $s = -Hg$. Davidon's [2] cubic interpolation technique is then applied until a point is found with $f^* < f$ and $\sigma' g^* > \sigma' g$. The point x^* is then accepted as the next point in the sequence.

(iii) Suppose that the step length α along the search direction s obtained by the previous step is "good"; then the distance $\alpha^* s^*$ to be moved from x^* will presumably be about the same as αs. Thus, before applying the above linear-search technique to estimate α, the length of the search vector s is scaled by $||s^{(k)}|| = ||\sigma^{(k-1)}||$, $k = 1, \ldots, n$. Numerical results indicate that this scaling criterion improves our algorithm for most problems tested.

This program is controlled by setting MODE = 1 in the parameter list of the subroutine MINI. For convenience, we shall denote this algorithm by MINI01. According to the results presented in [12], MINI01 appears preferable to MINI02 on functions where singularity of the true Hessian occurs.

Method 2. In this method we attempt to combine the merits of linear-search methods and step-length methods. The main feature of this algorithm is that linear-search techniques are applied infrequently. Using the notation previously developed, the algorithm can be described by the following seven major steps.

(1) *Create the search direction.* Evaluate $s^* = -H^* g^*$ and scale s^* such that $||s^*||$ is equal to $||\sigma||$ from the previous iteration for the first n iterations.

(2) *Perform the first extrapolation.* Estimate the first move by calculating

$$\alpha = \max\{-2(f - \bar{f})/s'g, 1\},$$

where $\bar{f}$ is the estimated least value of $f(x)$ provided by the user.

(3) *Evaluate the function value and its gradient.* Set $x^* = x + \alpha s$ and evaluate f^* and g^*. Set $\delta f = f^* - (f + \mu s'g)$, where $\mu = 0.0001$.

(4) *Test for termination of the linear search.* If $\delta f < 0$ and $s'g^* > s'g$, then the search for α is complete. Go to step (7).

(5) *Perform Shanno and Kettler's extrapolation technique.* If $\delta f < 0$ and $s'g^* < s'g$, there exists $\hat{\alpha}$ such that $\hat{\alpha} > \alpha$ and $f(x + \hat{\alpha}s) < f^*$. In this case, apply Shanno and Kettler's [10] extrapolation technique to estimate a new step length, β, that is,

$$\beta = \alpha^2 s'g / (2(f - f^* + \alpha s'g)).$$

Set $x = x^*$ and $\alpha = \beta$, and go to (3). Else go to step (6).

(6) *Perform Davidon's cubic interpolation technique.* If $\delta f \geq 0$ or $s'g^* \geq 0$, the minimum of $f(x)$ along s has been straddled. In this case, apply Davidon's [2] cubic interpolation technique to estimate a new step length, γ, that is,

$$\gamma = (-b + \sqrt{(b^2 - 3ac)})/(3a),$$

where

$$a = (2(f - f^*) + \alpha(s'g + s'g^*))/\alpha^3,$$

$$b = (-3(f - f^*) - \alpha(2s'g + s'g^*))/\alpha^2,$$

$$c = s'g.$$

If $b^2 - 3ac \leq 0$ or $| a | < 10^{-5} | b |$, then apply a quadratic interpolation scheme to estimate γ, that is,

$$\gamma = \alpha^2 s'g/(2(f - f^* + \alpha s'g)).$$

Now set $\alpha = \gamma$ and go to step (3).

(7) *Test for convergence.* If $|| g^* || \leq \epsilon$, then convergence is assumed. Otherwise update H^* by the BFS formula and repeat the whole process from step (1). Note that this convergence criterion has been found wanting in practice for badly scaled problems. Because it is the criterion used by VMM01 and MINFA, it was necessary to incorporate it in this program in order to obtain reasonable comparisons. A safer test can be incorporated by replacing the statement before statement 400 with

```
IF(DABS(G(I)).GT.EPS*(DABS(X(I)) + .001))CONV = .FALSE.
```

This changes the criterion from an absolute criterion on the gradient to a relative criterion, or the number of positions of accuracy required in the estimates to the minimum.

Remarks

(a) We note that the change in f on an iteration according to Taylor's theorem is approximately $s'g$ when s is small, but much less than $s'g$ in absolute value when the position of the minimum along a line is overestimated. The change in f relative to $s'g$ cannot become arbitrarily small if $(f^* - f)/s'g \geq \mu$, where $0 < \mu \ll 1$ is a preassigned small quantity. For more detail about the choice of this quantity, see Fletcher [3].

(b) On each move of the linear search, α is bounded in such a way that $\alpha s_i \leq \Delta x_i$, $i = 1, 2, \ldots, n$, where Δx is the vector of step bounds placed on all variables x_i. The step bounds Δx are introduced for several reasons. First, the initial estimated step length may be far too long, so that much is wasted in finding the desired α. Second, Δx helps in maintaining each component of x within the domain of definition of $f(x)$ so that environmental difficulties such as overflow and underflow problems may be prevented.

(c) In order to expedite the process of linear search, α is only estimated approximately. In fact, the search for α is complete as soon as the conditions given in step (4) are fulfilled. As a consequence, the new step length obtained by applying the extrapolation technique as described in step (5) is restricted to be $\alpha = \min\{\beta, 10\alpha\}$, so that the use of cubic interpolation technique may be avoided.

(d) In real situations, it may happen that the interpolation process gets stuck on one side of the minimum. To overcome this difficulty, the following modification to step (6) is suggested:

$$\alpha = \begin{cases} 0.1\ \alpha & \text{if } \gamma \leq 10^{-3}\alpha \\ \gamma & \text{if } 10^{-3}\alpha < \gamma \leq (1 - 10^{-3})\alpha \\ 0.8\ \alpha & \text{otherwise.} \end{cases}$$

(e) The use of this algorithm is signified by setting MODE = 2 in the parameter list of the subroutine MINI. For convenience, we shall denote this algorithm by MINI02. According to the results presented in [12], MINI02 is generally preferable to MINI01, except when the Hessian has singularities, as noted previously. For the general user, we would recommend use of MINI02, with a switch to MINI01 if convergence appears "too slow."

Program

The program consists of two modified quasi-Newton minimization techniques (MINI01, MINI02) which are controlled respectively by setting MODE = 1 or MODE = 2 in the parameter list of the subroutine MINI. This program requires one user-supplied subroutine (CALCFG), which calculates the values of the objective function and its first partial derivatives. The use of the subroutine and the

Table I. Comparison of Numerical Results Obtained by Four Algorithms

(n = number of function and gradient cells; γ = rank of each algorithm)

	MINI02		MINI01		MINFA		VMM01	
	n	γ	n	γ	n	γ	n	γ
BOX								
(5,0)	24	2	40	4	20	1	24	2
(0,0)	19	1	28	4	27	3	21	2
(0,20)	12	1	18	2	21	3	21	3
(2.5,10)	6	1	12	2	19	4	16	3
(5,20)	15	1	21	2	36	4	28	3
Total rank		6		14		16		14
ROSENBROCK								
(1,-1.2)	26	1	42	4	37	3	28	2
(2,-2)	18	1	34	3	39	4	20	2
(-3.635,5.621)	55	2	80	4	39	1	70	3
(6.39,-.221)	64	1	66	2	69	3	74	4
(1.489,-2.547)	24	1	37	3	43	4	28	2
Total rank		6		16		15		13
BEALE								
(1,-1.2)	13	1	23	4	15	2	16	3
(2,-2)	18	1	26	4	25	3	24	2
(0,0)	13	1	22	3	22	3	13	1
(5,3)	36	4	35	3	25	2	11	1
(10,10)	74	2	84	4	77	3	18	1
Total rank		9		18		14		9
WEIBULL								
(5,.15,2.5)	44	1	84	2	144	3	(3)	4+F
(250,.3,5)	63	1	94	4	81	2	93	3
(100,3,12.5)	48	1	66	2	74	3	(4)	4+F
Total rank		3		8		8		11+2F
WOOD								
(-3,-1,-3,-1)	99	2	91	1	113	3	132	4
(-3,1,-3,1)	98	2	78	1	100	3	153	4
(-1.2,1,-1.2,1)	84	2	54	1	112	4	95	3
(-1.2,1,1.2,1)	45	1	95	4	70	3	64	2
Total rank		7		7		13		13
POWELL								
(-3,-1,0,1)	39	1	57	3	(1)	4+F	47	2
CREGG								
(1,2,2,2)	45	1	79	2	(2)	3+F	(5)	3+F
Overall Total Rank		33		68		71+2F		63+3F

(1) = underflow occurred at the 73th iteration.
(2) = underflow occurred at the 1231th iteration.
(3) = the program exited with IEXIT=4, see Fletcher (4).
(4) = underflow occurred at the 2nd iteration.
(5) = the program exited with FEXIT=4, see Fletcher (4).

meaning of the parameters are described in the comments at the beginning of subroutine MINI. All communication between the main program and subroutines is achieved through the subroutine argument lists. An iteration is defined as the calculations required to select a new point which yields a lower function value than that of the previous one.

Test Results

These two programs (MINI01, MINI02) have been tested over a wide range of test functions in [12]. Compared with the subroutine (VMM01) given by Fletcher [3] and the subroutine (MINFA) as suggested by Powell [7], results show that MINI02 is the best algorithm, while MINI01 places second. For user comparisons,

we list some computational results of these two programs for the following four standard test functions.

(1) Box's function, defined by

$$f(x_1, x_2) = \sum_{i=1}^{10} \left| (e^{-x_1 t_i} - e^{-x_2 t_i}) - (e^{-t_i} - e^{-10 t_i}) \right|^2,$$

where t_i ranges from .1 to 1 in steps of .1. For MINI01 and MINI02, the step bounds are set to be $\mathrm{STP}(i) = 3$, $i = 1, 2$, and $\Delta^{(0)} = 3$ for MINFA.

(2) Rosenbrock's function, defined by

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2,$$

with $\mathrm{STP}(i) = 3$, $i = 1, 2$, and $\Delta^{(0)} = 3$ for MINFA.

(3) Wood's function, defined by

$$\begin{aligned} f(x_1, x_2, x_3, x_4) = {} & 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 \\ & + (1 - x_3)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] \\ & + 19.8(x_2 - 1)(x_4 - 1), \end{aligned}$$

with $\mathrm{STP}(i) = 10$, $i = 1, 2, 3, 4$, and $\Delta^{(0)} = 10$ for MINFA.

(4) Weibull's function, defined by

$$f(x_1, x_2, x_3) = \sum_{i=1}^{99} [\exp((x_3 - t_i)x^2/x_1) - y_i]^2,$$

where the y_i and t_i are perfect data generated for the 99 points corresponding to $y = .01$ to .99 in steps of .01, for the values $x_1 = 50$, $x_2 = 1.5$, $x_3 = 25$, and $\mathrm{STP}(1) = 300$, $\mathrm{STP}(2) = 3$, $\mathrm{STP}(3) = 30$, and $\Delta^{(0)} = 3$ for MINFA.

(5) Beale's function, defined by

$$f(x_1, x_2) = [1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2,$$

with $\Delta^{(0)} = 3$ for MINFA and $|\Delta x_i| \leq 3$, $i = 1, 2$, for MINI01 and MINI02.

(6) Powell's function, defined by

$$f(x_1, x_2, x_3, x_4) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4,$$

with $\Delta^{(0)} = 3$ for MINFA, and $|\Delta x_i| \leq 3$, $i = 1, 2, 3, 4$, for MINI01 and MINI02.

(7) Cregg's function, defined by

$$f(x_1, x_2, x_3, x_4) = (e^{x_1} - x_2)^4 + 100(x_2 - x_3)^6 + \tan^4(x_3 - x_4) + x_1^8 + (x_4 - 1)^2,$$

with $\Delta^{(0)} = 3$ for MINFA, and $|\Delta x_i| \leq 3$, $i = 1, 2, 3, 4$, for MINI01 and MINI02.

All tests were run in double precision on an IBM 370/165. In all cases, convergence was assumed to have occurred when $|\partial f/\partial x_i| \leq 10^{-5}$ for each i. Each test function was started from various initial estimates and their results are listed in Table I. The table does not differentiate function from gradient calls, because all programs tested calculate both each time the function subroutine is called. For a more complete evaluation, encompassing overhead and execution time, see [12]. We note here only that more careful evaluation leaves relative rankings unchanged.

REFERENCES

1. Broyden, C.G. The convergence of a class of double-rank minimization algorithms 2. The new algorithm. *JIMA 6* (1970), 222–231.
2. Davidon, W.C. Variable metric method for minimization. Rep. ANL-5990, Argonne Nat. Lab., Nov. 1959.
3. Fletcher, R. A new approach to variable metric algorithms. *Computer J. 13* (1970), 317–322.
4. Fletcher, R. A survey of algorithms for unconstrained optimization. Tech. Paper TP 456, Atomic Energy Research Establishment, Harwell, England, June 1971.
5. Goldfarb, D. A family of variable-metric methods derived by variational means, *Math. Computation 24* (1970), 23–26.
6. Greenstadt, J. Variations on variable-metric methods. *Math. Comp. 24* (1970), 7–18.
7. Powell, M.J.D. A fortran subroutine for unconstrained minimization, requiring first derivative of the objective function. Rep. AERE-R 6469, Atomic Energy Research Establishment, Harwell, England, 1970.

8. POWELL, M.J.D. Recent advances in unconstrained optimization. *Math. Programming 1* (1971), 26–57.
9. SHANNO, D.F. Conditioning of quasi-Newton methods for function minimization. *Math. Computation 24* (1970), 647–656.
10. SHANNO D.F. AND KETTLER, P.C. Optimal conditioning of quasi-Newton methods. *Math. Computation 24* (1970), 657–664.
11. SHANNO, D.F., BERG, A., AND CHESTON, G. Restarts and rotations of quasi-Newton methods. Information Processing 74, North-Holland Publ. Co., Amsterdam, 1974, pp. 557–561.
12. SHANNO, D.F., AND PHUA, K.H. Effective comparison of unconstrained optimization techniques. *Manage. Sci. 22* (1975), 321–330.

ALGORITHM

```
      SUBROUTINE MINI(N, X, F, G, H, M, IH, STP, EPS, FEST, MAXF,       MIN   10
     * MODE, ITER, IFUN, IERR, CALCFG)                                  MIN   20
C PURPOSE. THIS SUBROUTINE FINDS A MINIMUM OF THE FUNCTION F(X)         MIN   30
C -------  WITHIN THE ACCURACY  NORM(G(X)) .LE. EPS,  WHERE G(X) IS     MIN   40
C          THE GRADIENT OF F(X) AND EPS IS THE REQUIRED ACCURACY        MIN   50
C          PROVIDED BY THE USER.                                        MIN   60
C          THIS SUBROUTINE ADOPTS THE QUASI-NEWTON METHOD AS            MIN   70
C          SUGGESTED BY SHANNO, D.F. AND PHUA, K.H. (1974).             MIN   80
C DESCRIPTION OF PARAMETERS IN THE ARGUMENT LIST                        MIN   90
C ----------------------------------------------                        MIN  100
C N      NUMBER OF VARIABLES                                            MIN  110
C X      STORES THE CURRENT APPROXIMATION TO THE VARIABLES X. AN        MIN  120
C        INITIAL APPROXIMATION SHOULD BE PROVIDED ON ENTRY, WHICH       MIN  130
C        WILL BE REPLACED BY THE BEST ESTIMATE OBTAINED ON EXIT.        MIN  140
C F      ON RETURN, F CONTAINS THE CORRESPONDING VALUE OF F(X).         MIN  150
C G      ON RETURN, G IS THE CORRESPONDING GRADIENT VECTOR G(X).        MIN  160
C H      USED TO STORE THE APPROXIMATED INVERSE HESSIAN MATRIX AS       MIN  170
C        ROWS OF ITS UPPER TRIANGLE.                                    MIN  180
C M      = N*(N+1) / 2, THE LENGTH OF THE VECTOR H.                     MIN  190
C IH     INDICATOR CHOOSING THE INITIAL APPROXIMATION TO THE INVERSE    MIN  200
C        HESSIAN MATRIX,                                                MIN  210
C        = 0 THE MINI SUBROUTINE SETS H INITIALLY.                      MIN  220
C        = 1 IT IS ASSUMED THAT A GOOD ESTIMATE  OF THIS MATRIX IS      MIN  230
C            PROVIDED, FOR INSTANCE FROM PREVIOUS CALCULATIONS.         MIN  240
C STP    THE VECTOR CONTAINS THE MAXIMUM ALLOWABLE STEP-LENGTH ON       MIN  250
C        EACH COMPONENT OF X. THE USER MUST PROVIDE THIS VECTOR.        MIN  260
C        PROPER CHOICE OF THE COMPONENTS STP(I) IS IMPORTANT FOR        MIN  270
C        EFFICIENT EXECUTION. IN GENERAL, CHOOSING STP(I) TOO LARGE     MIN  280
C        IS PREFERABLE TO CHOOSING STP(I) TOO SMALL. THUS STP(I)        MIN  290
C        SHOULD IDEALLY BE CHOSEN TO LIMIT SEARCH TO THE SMALLEST       MIN  300
C        REGION ABOUT X(I) IN WHICH THE USER HAS CONFIDENCE THE TRUE    MIN  310
C        OPTIMUM LIES. IF THE USER HAS NO FEEL FOR THE LOCATION OF      MIN  320
C        THE TRUE OPTIMUM, STP(I) SHOULD BE CHOSEN TO BE LARGE,         MIN  330
C        REFLECTING THE USERS VIEW THAT THE OPTIMUM MAY BE FAR FROM     MIN  340
C        THE INITIAL ESTIMATES.                                         MIN  350
C EPS    ACCURACY REQUIRED IN EACH ELEMENT OF THE GRADIENT VECTOR       MIN  360
C        G(X). IF THE USER HAS NO FEEL FOR THE DESIRED ACCURACY OF      MIN  370
C        HIS SOLUTION, EPS = 10**(-5) IS SUGGESTED.                     MIN  380
C FEST   A LOWER BOUND ON THE VALUE OF F PROVIDED BY THE USER.          MIN  390
C MAXF   MAXIMUM ALLOWABLE FUNCTION EVALUATIONS, PROVIDED BY THE        MIN  400
C        USER. MAXF ACTS SOLELY AS A DEVICE TO LIMIT EXECUTION TIME     MIN  410
C        IN THE EVENT OF SLOW CONVERGENCE. IF THE USER HAS NO FEEL      MIN  420
C        FOR EXECUTION TIME AS A FUNCTION OF TIME/ITERATION,            MIN  430
C        MAXF = 200 IS SUGGESTED.                                       MIN  440
C MODE   ROUTINE SELECTOR PROVIDED BY THE USER,                         MIN  450
C        = 1  MEANS THE ROUTINE MINI01 WILL BE USED.                    MIN  460
C        = 2  MEANS THE ROUTINE MINI02 WILL BE USED.                    MIN  470
C        MODE = 2 IS SUGGESTED UNLESS CONVERGENCE IS UNACCEPTABLY       MIN  480
C        SLOW. IN THIS CASE, MODE = 1 SHOULD BE TRIED.                  MIN  490
C ITER   NUMBER OF ITERATIONS USED.                                     MIN  500
C IFUN   NUMBER OF FUNCTION EVALUATIONS USED.                           MIN  510
C IERR   ERROR MESSAGE RETURNED,                                        MIN  520
C        = 0 MEANS NO ERROR ENCOUNTED.                                  MIN  530
C        = 1 MEANS IT REQUIRES  MORE FUNCTION EVALUATIONS.              MIN  540
C        = 2 MEANS INCREMENT IN X IS TOO SMALL, PROGRAM STUCK,          MIN  550
C            POSSIBLY CAUSED BY  1) H SINGULAR, 2) EPS SET TOO SMALL    MIN  560
C            (SEEK EXPERT FOR ADVICE).                                  MIN  570
C        = 3 MEANS BOTH THE SEARCH DIRECTION AND ITS REVERSE            MIN  580
C            DIRECTION ARE NOT DOWN-HILL, POSSIBLY CAUSED BY  G         MIN  590
C            PROGRAMMED INCORRECTLY.                                    MIN  600
C        = 4 MEANS THAT FUNCTION VALUE LESS THAN FEST DETECTED.         MIN  610
C REMARKS                                                               MIN  620
C -------                                                               MIN  630
C 1) THE USER MUST PROVIDE A SUBROUTINE                                 MIN  640
C              SUBROUTINE CALCFG (N, X, F, G)                           MIN  650
C              DOUBLE PRECISION X(N), G(N), F                           MIN  660
C    WHICH, GIVEN A VECTOR X IN X(1), X(2),..., X(N), CALCULATES THE    MIN  670
C    FUNCTION F(X) AND THE GRADIENT G(X) AND PLACES THEM IN F AND       MIN  680
C    G(1), G(2), ..., G(N) RESPECTIVELY.                                MIN  690
C 2) THE VECTOR D(10) IS USED TO HOLD THE CURRENT SEARCH DIRECTION,     MIN  700
C    WHILE XX(10) AND GG(10) ARE USED TO STORE THE PREVIOUS VALUES      MIN  710
C    OF X AND G. THUS IF THE VALUE OF N IS GREATER THAN 10, THEN        MIN  720
C    THEIR DIMENSIONS MUST BE CHANGED ACCORDINGLY.                      MIN  730
```

```
      DOUBLE PRECISION F, FEST, EPS, X(N), G(N), STP(N), H(M)          MIN  740
      DOUBLE PRECISION D(10), XX(10), GG(10), EPS2, DUM, STEP, SUM,    MIN  750
     * G1, G2                                                          MIN  760
      DOUBLE PRECISION STPMAX, ALPHA, AL, BL, FG, F1, FM, GM, AA,     MIN  770
     * BB, CC                                                          MIN  780
      DOUBLE PRECISION SS, DY, YHY, C1, C2, DSQRT, DABS                MIN  790
      LOGICAL CONV                                                     MIN  800
C DO INITIALIZATION OF PARAMETERS, STEP-BOUND AND UPDATE               MIN  810
      CALL CALCFG(N, X, F, G)                                          MIN  820
      IFUN = 1                                                         MIN  830
      ITER = 0                                                         MIN  840
      IERR = 0                                                         MIN  850
      EPS2 = EPS*EPS                                                   MIN  860
C CALCULATE THE INITIAL STEP-BOUND                                     MIN  870
      DUM = 0.                                                         MIN  880
      STEP = 0.                                                        MIN  890
      DO 10 I=1,N                                                      MIN  900
        DUM = DUM + G(I)*G(I)                                          MIN  910
        STEP = STEP + STP(I)*STP(I)                                    MIN  920
   10 CONTINUE                                                         MIN  930
      IF (DUM.EQ.0.) RETURN                                            MIN  940
C SET UP THE INITIAL UPDATE                                            MIN  950
      IF (IH.EQ.0) GO TO 20                                            MIN  960
      GO TO 50                                                         MIN  970
C THEN                                                                 MIN  980
   20 DUM = 100.*DSQRT(STEP/DUM)                                       MIN  990
      IJ = 1                                                           MIN 1000
      DO 40 I=1,N                                                      MIN 1010
        DO 30 J=I,N                                                    MIN 1020
          H(IJ) = 0.                                                   MIN 1030
          IF (I.EQ.J) H(IJ) = DUM                                      MIN 1040
          IJ = IJ + 1                                                  MIN 1050
   30   CONTINUE                                                       MIN 1060
   40 CONTINUE                                                         MIN 1070
   50 CONTINUE                                                         MIN 1080
C REPEAT THE FOLLOWING 5 STEPS UNTIL EXIT FROM 4TH STEP                MIN 1090
C    ***************************************************************   MIN 1100
C    * STEP 1. SET UP THE SEARCH DIRECTION AND SCALE ITS LENGTH    *   MIN 1110
C    ***************************************************************   MIN 1120
   60 SUM = 0.                                                         MIN 1130
      DO 110 I=1,N                                                     MIN 1140
        DUM = 0.                                                       MIN 1150
        IJ = I                                                         MIN 1160
        IF (I.GT.1) GO TO 70                                           MIN 1170
        GO TO 90                                                       MIN 1180
C THEN                                                                 MIN 1190
   70   II = I - 1                                                     MIN 1200
        DO 80 J=1,II                                                   MIN 1210
          DUM = DUM - H(IJ)*G(J)                                       MIN 1220
          IJ = IJ + N - J                                              MIN 1230
   80   CONTINUE                                                       MIN 1240
   90   CONTINUE                                                       MIN 1250
        DO 100 J=I,N                                                   MIN 1260
          DUM = DUM - H(IJ)*G(J)                                       MIN 1270
          IJ = IJ + 1                                                  MIN 1280
  100   CONTINUE                                                       MIN 1290
        D(I) = DUM                                                     MIN 1300
        XX(I) = X(I)                                                   MIN 1310
        GG(I) = G(I)                                                   MIN 1320
        SUM = SUM + DUM*DUM                                            MIN 1330
  110 CONTINUE                                                         MIN 1340
      IF ((ITER.GE.1) .AND. (ITER.LE.N)) GO TO 120                     MIN 1350
      GO TO 140                                                        MIN 1360
C THEN - SCALE THE LENGTH OF THE SEARCH VECTOR                         MIN 1370
  120 DUM = DSQRT(STEP/SUM)                                            MIN 1380
      DO 130 I=1,N                                                     MIN 1390
        D(I) = D(I)*DUM                                                MIN 1400
  130 CONTINUE                                                         MIN 1410
  140 CONTINUE                                                         MIN 1420
C ***************************************************************    MIN 1430
C * STEP 2. TEST IF THE SEARCH DIRECTION IS DOWNHILL             *    MIN 1440
C ***************************************************************    MIN 1450
      DO 170 J=1,2                                                     MIN 1460
        G1 = 0.                                                        MIN 1470
        DO 150 I=1,N                                                   MIN 1480
          G1 = G1 + D(I)*G(I)                                          MIN 1490
  150   CONTINUE                                                       MIN 1500
        IF (G1.LT.0.) GO TO 180                                        MIN 1510
        DO 160 I=1,N                                                   MIN 1520
          D(I) = -D(I)                                                 MIN 1530
  160   CONTINUE                                                       MIN 1540
  170 CONTINUE                                                         MIN 1550
  180 IF (G1.GT.0.) GO TO 190                                          MIN 1560
      GO TO 200                                                        MIN 1570
C THEN                                                                 MIN 1580
  190 IERR = 3                                                         MIN 1590
C ........ERROR EXIT - BOTH D AND -D ARE NOT DOWNHILL                  MIN 1600
      RETURN                                                           MIN 1610
  200 CONTINUE                                                         MIN 1620
C FIND OUT THE MAXIMUM ALLOWABLE STEP LENGTH, STPMAX                   MIN 1630
      STPMAX = 1.0D+30                                                 MIN 1640
```

```
      DO 210 I=1,N                                                      MIN 1650
        IF (DABS(D(I)).LT.1.D-30) GO TO 210                             MIN 1660
        DUM = DABS(STP(I)/D(I))                                         MIN 1670
        IF (STPMAX.GT.DUM) STPMAX = .9*DUM                              MIN 1680
  210 CONTINUE                                                          MIN 1690
C ***************************************************************     MIN 1700
C * STEP 3. PERFORM THE LINEAR SEARCH                             *     MIN 1710
C ***************************************************************     MIN 1720
      ALPHA = 0.                                                        MIN 1730
      FG = G1                                                           MIN 1740
      AL = 1.                                                           MIN 1750
      IF (MODE.EQ.2) AL = 2.*(FEST-F)/G1                                MIN 1760
      IF (AL.GT.1.) AL = 1.                                             MIN 1770
      IF (AL.GT.STPMAX) AL = 0.8*STPMAX                                 MIN 1780
      IF (AL.LT.0.) GO TO 220                                           MIN 1790
      GO TO 230                                                         MIN 1800
C THEN                                                                  MIN 1810
  220 IERR = 4                                                          MIN 1820
C .....ERROR EXIT - F .LT. FEST PROVIDED BY THE USER                    MIN 1830
      RETURN                                                            MIN 1840
  230 CONTINUE                                                          MIN 1850
C REPEAT THE FOLLOWING EXTRAPOLATION PROCESS                            MIN 1860
  240 F1 = F                                                            MIN 1870
      ALPHA = ALPHA + AL                                                MIN 1880
      DO 250 I=1,N                                                      MIN 1890
        X(I) = XX(I) + ALPHA*D(I)                                       MIN 1900
  250 CONTINUE                                                          MIN 1910
      CALL CALCFG(N, X, F, G)                                           MIN 1920
      IFUN = IFUN + 1                                                   MIN 1930
      IF (IFUN.GT.MAXF) GO TO 260                                       MIN 1940
      GO TO 270                                                         MIN 1950
C THEN                                                                  MIN 1960
  260 IERR = 1                                                          MIN 1970
C .....ERROR EXIT - EXCEED THE MAXIMUM ALLOWED FUNCTION CALLS           MIN 1980
      RETURN                                                            MIN 1990
  270 CONTINUE                                                          MIN 2000
      G2 = 0.0                                                          MIN 2010
      DO 280 I=1,N                                                      MIN 2020
        G2 = G2 + G(I)*D(I)                                             MIN 2030
  280 CONTINUE                                                          MIN 2040
      CONV = (G2.GT.FG) .AND. (F1.GT.(F+.0001*G1))                      MIN 2050
C ...EXIT THE EXTRAPOLATION LOOP, IF THE MINIMUM HAS BEEN FOUND         MIN 2060
      IF (CONV .AND. (MODE.EQ.2)) GO TO 290                             MIN 2070
C ...EXIT THE EXTRAPOLATION LOOP, IF THE MINIMUM IS STRADDLED           MIN 2080
      IF ((F.GT.F1) .OR. (G2.GT.0.)) GO TO 290                          MIN 2090
      BL = AL                                                           MIN 2100
      AL = 0.5*G1*BL*BL/(F1-F+BL*G1)                                    MIN 2110
      IF ((MODE.EQ.2) .AND. (AL.LT.BL)) AL = 2.*(FEST-F)/G1             MIN 2120
      IF (AL.GT.(10.*BL)) AL = 10.*BL                                   MIN 2130
      IF (AL.GT.STPMAX) AL = 0.8*STPMAX                                 MIN 2140
      IF (AL.LT.(1.001*BL)) AL = BL + BL                                MIN 2150
      G1 = G2                                                           MIN 2160
C *GO BACK TO THE EXTRAPOLATION PROCESS                                 MIN 2170
      GO TO 240                                                         MIN 2180
  290 CONTINUE                                                          MIN 2190
      IF (.NOT.(CONV) .OR. (MODE.EQ.1)) GO TO 300                       MIN 2200
      GO TO 390                                                         MIN 2210
C THEN                                                                  MIN 2220
C    REPEAT THE FOLLOWING INTERPOLATION PROCESS                         MIN 2230
  300 BL = AL                                                           MIN 2240
      AA = (G1+G2+2.*(F1-F)/AL)/(AL*AL)                                 MIN 2250
      BB = (G2-3.*AA*AL*AL-G1)/(AL+AL)                                  MIN 2260
      CC = BB*BB - 3.*AA*G1                                             MIN 2270
      DUM = DABS(AA)*1.0D+05 - DABS(BB)                                 MIN 2280
      AL = (.5*G1*BL*BL)/(F1-F+BL*G1)                                   MIN 2290
      IF ((CC.GT.0.) .AND. (DUM.GT.0.)) AL = (-BB+DSQRT(CC))/(3.*AA)    MIN 2300
      FM = F                                                            MIN 2310
      GM = G2                                                           MIN 2320
      IF (AL.LE.(.001*BL)) AL = .1*BL                                   MIN 2330
      IF (AL.GT.(.999*BL)) AL = .8*BL                                   MIN 2340
      ALPHA = ALPHA - BL + AL                                           MIN 2350
      DO 310 I=1,N                                                      MIN 2360
        X(I) = XX(I) + ALPHA*D(I)                                       MIN 2370
  310 CONTINUE                                                          MIN 2380
      CALL CALCFG(N, X, F, G)                                           MIN 2390
      IFUN = IFUN + 1                                                   MIN 2400
      IF (IFUN.GT.MAXF) GO TO 320                                       MIN 2410
      GO TO 330                                                         MIN 2420
C THEN                                                                  MIN 2430
  320 IERR = 1                                                          MIN 2440
C .............ERROR EXIT - EXCEED THE MAXIMUM ALLOWED FUNCTION       MIN 2450
C CALLS                                                                 MIN 2460
      RETURN                                                            MIN 2470
  330 CONTINUE                                                          MIN 2480
      G2 = 0.                                                           MIN 2490
      DO 340 I=1,N                                                      MIN 2500
        G2 = G2 + D(I)*G(I)                                             MIN 2510
  340 CONTINUE                                                          MIN 2520
      CONV = (G2.GT.FG) .AND. (F1.GT.(F+.0001*G1))                      MIN 2530
C ...EXIT THE INTERPOLATION LOOP, IF THE MINIMUM IS FOUND               MIN 2540
      IF (CONV) GO TO 380                                               MIN 2550
```

```
C ...EXIT THE INTERPOLATION LOOP, IF STEP-SIZE IS TOO SMALL          MIN 2560
      IF ((G2.GT.FG) .AND. (BL.LE.EPS2)) GO TO 380                   MIN 2570
      SS = 0.                                                        MIN 2580
      DO 350 I=1,N                                                   MIN 2590
        SS = SS + DABS(G(I)*D(I))                                    MIN 2600
  350 CONTINUE                                                       MIN 2610
      SS = SS/(100000.*FLOAT(N))                                     MIN 2620
C ...EXIT THE INTERPOLATION LOOP, IF TWO POINTS ARE CLOSED           MIN 2630
      IF ((G2.LT.SS) .AND. (G2.GT.FG)) GO TO 380                     MIN 2640
      IF ((G2.LT.0.) .AND. (F1.GT.F)) GO TO 360                      MIN 2650
      GO TO 370                                                      MIN 2660
C THEN - ADVANCE TO THE LOWER POINT                                  MIN 2670
  360 G1 = G2                                                        MIN 2680
      G2 = GM                                                        MIN 2690
      F1 = F                                                         MIN 2700
      F = FM                                                         MIN 2710
      AL = BL - AL                                                   MIN 2720
      ALPHA = ALPHA + AL                                             MIN 2730
  370 CONTINUE                                                       MIN 2740
C *GO BACK TO THE INTERPOLATION PROCESS                              MIN 2750
      GO TO 300                                                      MIN 2760
  380 CONTINUE                                                       MIN 2770
  390 CONTINUE                                                       MIN 2780
C ***************************************************************   MIN 2790
C * STEP 4. TEST FOR CONVERGENCE                                *   MIN 2800
C ***************************************************************   MIN 2810
      CONV = .TRUE.                                                  MIN 2820
      STEP = 0.                                                      MIN 2830
      DO 400 I=1,N                                                   MIN 2840
        D(I) = X(I) - XX(I)                                          MIN 2850
        STEP = STEP + D(I)*D(I)                                      MIN 2860
        IF ((G(I)*G(I)).GT.EPS2) CONV = .FALSE.                      MIN 2870
  400 CONTINUE                                                       MIN 2880
C .....NORMAL EXIT                                                   MIN 2890
      IF (CONV) RETURN                                               MIN 2900
      IF (STEP.LE.(EPS2*EPS2)) GO TO 410                             MIN 2910
      GO TO 420                                                      MIN 2920
C THEN                                                               MIN 2930
  410 IERR = 2                                                       MIN 2940
C ........ERROR EXIT - THE STEP SIZE IS TOO SMALL                    MIN 2950
      RETURN                                                         MIN 2960
  420 CONTINUE                                                       MIN 2970
C ***************************************************************   MIN 2980
C * STEP 5. UPDATE THE APPROXIMATED INVERSE HESSIAN MATRIX      *   MIN 2990
C ***************************************************************   MIN 3000
      DY = 0.                                                        MIN 3010
      YHY = 0.                                                       MIN 3020
      DO 430 I=1,N                                                   MIN 3030
        GG(I) = G(I) - GG(I)                                         MIN 3040
        DY = DY + D(I)*GG(I)                                         MIN 3050
  430 CONTINUE                                                       MIN 3060
      DO 480 I=1,N                                                   MIN 3070
        DUM = 0.                                                     MIN 3080
        IJ = I                                                       MIN 3090
        IF (I.GT.1) GO TO 440                                        MIN 3100
        GO TO 460                                                    MIN 3110
C THEN                                                               MIN 3120
  440   II = I - 1                                                   MIN 3130
        DO 450 J=1,II                                                MIN 3140
          DUM = DUM + H(IJ)*GG(J)                                    MIN 3150
          IJ = IJ + N - J                                            MIN 3160
  450   CONTINUE                                                     MIN 3170
  460   CONTINUE                                                     MIN 3180
        DO 470 J=I,N                                                 MIN 3190
          DUM = DUM + H(IJ)*GG(J)                                    MIN 3200
          IJ = IJ + 1                                                MIN 3210
  470   CONTINUE                                                     MIN 3220
        YHY = YHY + DUM*GG(I)                                        MIN 3230
        XX(I) = DUM                                                  MIN 3240
  480 CONTINUE                                                       MIN 3250
      C1 = 1. + YHY/DY                                               MIN 3260
      DO 490 I=1,N                                                   MIN 3270
        GG(I) = C1*D(I) - XX(I)                                      MIN 3280
  490 CONTINUE                                                       MIN 3290
      IJ = 1                                                         MIN 3300
      DO 510 I=1,N                                                   MIN 3310
        C1 = D(I)/DY                                                 MIN 3320
        C2 = XX(I)/DY                                                MIN 3330
        DO 500 J=I,N                                                 MIN 3340
          H(IJ) = H(IJ) + C1*GG(J) - C2*D(J)                         MIN 3350
          IJ = IJ + 1                                                MIN 3360
  500   CONTINUE                                                     MIN 3370
  510 CONTINUE                                                       MIN 3380
      ITER = ITER + 1                                                MIN 3390
C *STEP 5 NOW COMPLETE - GO BACK TO STEP 1                           MIN 3400
      GO TO 60                                                       MIN 3410
C *END OF REPEAT BLOCK                                               MIN 3420
C CONTINUE                                                           MIN 3430
      END                                                            MIN 3440
```

ACM Transactions on Mathematical Software, Vol. 3, No. 1, March 1977. Page—112

REMARK ON ALGORITHM 500

Minimization of Unconstrained Multivariate Functions [E4]
[D.F. Shanno and K.H. Phua, *ACM Trans. Math. Software 2*, 1 (March 1976), 87–94]

Charles Dunham [Recd June 7, 1976]
Computer Science Department, University of Western Ontario, London, Ont., Canada.

The arrays STP and H used as arguments to this algorithm must be declared double precision, as declared in the body (but not the published heading) of the subroutine MINI.

MINI was used to determine a best $l_{1.5}$ approximation by the approximating function

$$F(A, x) = a_1 + a_2x + a_3 \exp(a_4x)$$

on $\{0, 1/10, \ldots, 9/10, 1\}$ to the function $f(x) = x + \exp(x) + \exp(2x)/100$. This is a minimization problem with four parameters, a_1, a_2, a_3, a_4. When MINI was run initially with arguments STP single precision of dimension 4 and H single precision of dimension 10 on a CDC Cyber 73, the program hung with an error mode 2 (infinite operand). With the above change (to double precision) the program produced the approximation obtained independently by several other methods.

ACM Transactions on Mathematical Software, Vol. 6, No. 4, December 1980, Pages 618–622.

REMARK ON ALGORITHM 500

Minimization of Unconstrained Multivariate Functions [E4] [D.F. Shanno and K.H. Phua, *ACM Trans. Math. Softw. 2*, 1 (March 1976), 87–94.]

D.F. Shanno and K.H. Phua [Received 3 January 1979; revised 5 June 1979; accepted 26 December 1979]
D.F. Shanno, Department of Management Information Systems, College of Business and Public Administration, The University of Arizona, Tucson, AZ 85721; K.H. Phua, Department of Computer Science, Nanyang University, Upper Jurong Road, Singapore 22, Republic of Singapore.

DESCRIPTION

1. Purpose

This subroutine finds a local minimizer of a nonlinear function of n variables $f(x)$ where $x = (x_1, \ldots, x_n)$, $n \geq 1$ can be any real numbers. This algorithm is meant to supersede the algorithm documented in [11].

2. Methods

The subroutine incorporates two nonlinear optimization methods, a conjugate gradient algorithm and a variable metric algorithm, with the choice of method left to the user.

The conjugate gradient algorithm is the Beale restarted memoryless variable metric algorithm documented in Shanno [7]. This method requires approximately $7n$ double-precision words of working storage to be provided by the user. The variable metric method is the BFGS algorithm with initial scaling documented in Shanno and Phua [10], and required approximately $n^2/2 + 11n/2$ double-precision words of working storage.

Whichever method is chosen, the same linear search technique is used for both methods, with two differences. The basic linear search uses Davidon's cubic interpolation to find a step length α, which satisfies

$$f(x + \alpha d) < f(x) + \alpha d' g(x)\ 0.0001, \tag{1}$$

where d is the chosen search direction, $g(x) = \nabla f(x)$, the gradient of f at x, and $d'g(x)$, the directional derivative of $f(x)$ at x along d, is always a negative number. In addition, α must satisfy

$$| d'g(x + \alpha d)/d'g(x) | < 0.9. \tag{2}$$

The convergence of the BFGS variable metric algorithm under the conditions (1) and (2) has been studied by Powell [6], while the convergence of the conjugate gradient method has been studied by Shanno [8].

The major difference between the two methods insofar as the linear search is concerned is that if the first trial α satisfies (1) and (2), it is accepted if a variable metric method is used, but at least two trial α's are required before accepting an α satisfying (1) and (2) for the conjugate gradient method. Reasons for requiring this are detailed in Shanno [9].

The second difference between the two methods is that for a variable metric method, $\alpha = 1$ is always the initial α tried, while for the conjugate gradient method, $\alpha = 1$ is tried only for restart iterations, whereas for nonrestart iterations the initial α at step $k + 1$, denoted by α_{k+1}, is chosen to be $\alpha_{k+1} = \alpha_k\, d'_k g_k / d'_{k+1} g_{k+1}$, where d_k and g_k and d_{k+1} and g_{k+1} are the search vectors and gradients at the kth and $k + 1$st points, respectively.

The linear search contains safeguards to ensure that the search procedure cannot become stuck or attempt to move past a local maximum to a more distant local minimum.

Convergence is determined to have occurred when $\| g \| < \epsilon \max(1, \| x \|)$, where $\| \cdot \|$ is the Euclidean norm and ϵ is user supplied.

3. Test Results

Table I contains test results for a variety of test problems for both the BFGS and conjugate gradient method. The test functions are Wood's function for the listed initial estimates, the extended Rosenbrock function documented in [7], Brodlie's [1] variable-dimensioned Watson function, Oren's [5] power function, Powell's four-dimensioned function [6], Fletcher and Powell's trigonometric functions with initial estimates as in [2], and for various dimensions, the Mancino function with initial estimates as documented in [7], Moré, Garbow, and Hillstrom's [4] boundary-value problem, and Toint's variation on Broyden's function [12]. The Wood and Powell functions are documented in [3].

In the table, ITER represents the number of search directions calculated, while IFUN represents the number of function and gradient evaluations that were performed. In all cases, $\epsilon = 10^{-5}$ was used, except for the boundary value problem, where $\epsilon = 10^{-4}$.

With the exception of the power function, on which the BFGS does not perform well for reasons documented in [10], it is clear from Table I that the variable metric method is quite a bit more efficient in terms of function and gradient calls, primarily due to the fact that the first trial α can be accepted, while at least two trial α's per iteration must be tried by the conjugate gradient algorithm.

In terms of execution time, however, the issue is not so clear-cut. On a DEC-10 computer, the Broyden–Toint function with $n = 30$ took 6.49 CPU seconds for the BFGS, while the conjugate gradient method took 5.32 due to the overhead of updating the approximate Hessian at each step. However, for the 30-variable Mancino function, the BFGS took 28.12 CPU seconds, while the conjugate gradient method took 72.31. As one would expect, evaluations of the Mancino function are quite expensive, while the Broyden–Toint function evaluations are quite inexpensive.

Thus for large problems where space limitations do not preclude using the BFGS algorithm, users are urged to experiment to determine the most efficient algorithm for a particular problem. For small problems, we recommend the BFGS method, while for very large problems, memory considerations generally mandate using the conjugate gradient algorithm.

Table I

	BFGS		Conjugate Gradient	
	ITER	IFUN	ITER	IFUN
WOOD ($n = 4$)				
$-3, -1, -3, -1$	36	43	48	106
$-3, 1, -3, 1$	91	114	90	210
$-1.2, 1, -1.2, 1$	87	107	77	181
$-1.2, 1, 1.2, 1$	48	57	46	100
EROSEN				
$-1.2, 1, 1, 1, 1$ ($n = 5$)	117	150	132	278
$-1, \ldots, -1$ ($n = 10$)	674	893	946	1940
WATSON				
$0, \ldots, 0$ ($n = 5$)	37	39	34	69
$0, \ldots, 0$ ($n = 10$)	92	95	179	360
POWER				
$1, \ldots, 1$ ($n = 20$)	280	281	16	33
$1, \ldots, 1$ ($n = 50$)	539	540	30	61
POWELL ($n = 4$)				
$-3, -1, 0, 1$	48	49	28	57
TRIG				
($n = 5$)	20	22	20	41
($n = 10$)	35	37	44	89
($n = 15$)	57	59	100	201
MANCINO				
($n = 10$)	9	10	12	28
($n = 20$)	10	14	14	33
($n = 30$)	11	17	18	49
BOUNDARY VALUE				
($n = 10$)	28	30	25	51
($n = 20$)	56	58	48	97
($n = 30$)	86	88	121	243
BROYDEN-TOINT				
$-1, \ldots, -1$ ($n = 10$)	27	28	23	47
$-1, \ldots, -1$ ($n = 20$)	36	37	36	73
$-1, \ldots, -1$ ($n = 30$)	47	48	46	93

ACKNOWLEDGMENT

The authors are deeply indebted to Professor M.J.D. Powell for many useful suggestions concerning the linear search method adopted and for illustrating several adopted methods for coding optimization routines.

REFERENCES

1. BRODLIE, K.W. An assessment of two approaches to variable metric methods. *Math. Program. 12* (1977), 334–355.
2. FLETCHER, R., AND POWELL, M.J.D. A rapidly convergent descent method for minimization. *Comput. J. 6* (1963), 163–168.
3. HIMMELBLAU, D.M. *Applied Nonlinear Programming.* McGraw-Hill, New York, 1972.
4. MORÉ, J.J., GARBOW, B.S., AND HILLSTROM, K.E. Testing unconstrained minimization software. TM-324, Appl. Math. Div., Argonne National Lab., Argonne, Ill., 1978.
5. OREN, S.S. Self-scaling variable metric algorithm. Part II. *Manage. Sci. 20* (1974), 863–874.
6. POWELL, M.J.D. Some global convergence properties of a variable metric algorithm for minimization without exact line searches. In *Nonlinear Programming,* R.W. Cottle and C.E. Lemke, Eds., American Mathematical Soc., Providence, R.I., 1976.
7. SHANNO, D.F. Conjugate gradient methods with inexact searches. *Math. Oper. Res. 3* (1978), 244–256.
8. SHANNO, D.F. On the convergence of a new conjugate gradient method. *SIAM. J. Numer. Anal. 15* (1978), 1247–1257.
9. SHANNO, D.F. Quadratic termination of conjugate gradient methods. In *Extrenal Methods and Systems Analysis,* A.V. Fiacco and K.O. Kortanek, Eds., Springer-Verlag, New York, 1980, pp. 433–441.
10. SHANNO, D.F., AND PHUA, K.H. Matrix conditioning and nonlinear optimization. *Math. Program. 14* (1978), 149–160.
11. SHANNO, D.F., AND PHUA, K.H. Algorithm 500. Minimization of unconstrained multivariate functions. *ACM Trans. Math. Softw. 2,* 1 (1976), 87–94.
12. TOINT, PH.L. Some numerical results using a sparse matrix updating formula in unconstrained optimization. *Math. Comput. 32* (1978), 839–852.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service.]

```
      SUBROUTINE CONMIN(N,X,F,G,IFUN,ITER,EPS,NFLAG,MXFUN,W,
     1IOUT,MDIM,IDEV,ACC,NMETH)
C
C PURPOSE:     SUBROUTINE CONMIN MINIMIZES AN UNCONSTRAINED NONLINEAR
C              SCALAR VALUED FUNCTION OF A VECTOR VARIABLE X
C              EITHER BY THE BFGS VARIABLE METRIC ALGORITHM OR BY A
C              BEALE RESTARTED CONJUGATE GRADIENT ALGORITHM.
C
C USAGE:       CALL CONMIN(N,X,F,G,IFUN,ITER,EPS,NFLAG,MXFUN,W,
C              IOUT,MDIM,IDEV,ACC,NMETH)
C
C PARAMETERS: N       THE NUMBER OF VARIABLES IN THE FUNCTION TO
C                     BE MINIMIZED.
C             X       THE VECTOR CONTAINING THE CURRENT ESTIMATE TO
C                     THE MINIMIZER. ON ENTRY TO CONMIN,X MUST CONTAIN
C                     AN INITIAL ESTIMATE SUPPLIED BY THE USER.
C                     ON EXITING,X WILL HOLD THE BEST ESTIMATE TO THE
C                     MINIMIZER OBTAINED BY CONMIN. X MUST BE DOUBLE
C                     PRECISIONED AND DIMENSIONED N.
C             F       ON EXITING FROM CONMIN,F WILL CONTAIN THE LOWEST
C                     VALUE OF THE OBJECT FUNCTION OBTAINED.
C                     F IS DOUBLE PRECISIONED.
C             G       ON EXITING FROM CONMIN,G WILL CONTAIN THE
C                     ELEMENTS OF THE GRADIENT OF F EVALUATED AT THE
C                     POINT CONTAINED IN X. G MUST BE DOUBLE
C                     PRECISIONED AND DIMENSIONED N.
C             IFUN    UPON EXITING FROM CONMIN,IFUN CONTAINS THE
C                     NUMBER OF TIMES THE FUNCTION AND GRADIENT
C                     HAVE BEEN EVALUATED.
C             ITER    UPON EXITING FROM CONMIN,ITER CONTAINS THE
C                     TOTAL NUMBER OF SEARCH DIRECTIONS CALCULATED
C                     TO OBTAIN THE CURRENT ESTIMATE TO THE MINIZER.
C             EPS     EPS IS THE USER SUPPLIED CONVERGENCE PARAMETER.
C                     CONVERGENCE OCCURS WHEN THE NORM OF THE GRADIENT
C                     IS LESS THAN OR EQUAL TO EPS TIMES THE MAXIMUM
C                     OF ONE AND THE NORM OF THE VECTOR X. EPS
C                     MUST BE DOUBLE PRECISIONED.
C             NFLAG   UPON EXITING FROM CONMIN,NFLAG STATES WHICH
C                     CONDITION CAUSED THE EXIT.
C                     IF NFLAG=Ø, THE ALGORITHM HAS CONVERGED.
C                     IF NFLAG=1, THE MAXIMUM NUMBER OF FUNCTION
C                        EVALUATIONS HAVE BEEN USED.
C                     IF NFLAG=2, THE LINEAR SEARCH HAS FAILED TO
C                        IMPROVE THE FUNCTION VALUE. THIS IS THE
C                        USUAL EXIT IF EITHER THE FUNCTION OR THE
C                        GRADIENT IS INCORRECTLY CODED.
C                     IF NFLAG=3, THE SEARCH VECTOR WAS NOT
C                        A DESCENT DIRECTION. THIS CAN ONLY BE CAUSED
C                        BY ROUNDOFF,AND MAY SUGGEST THAT THE
C                        CONVERGENCE CRITERION IS TOO STRICT.
C             MXFUN   MXFUN IS THE USER SUPPLIED MAXIMUM NUMBER OF
C                     FUNCTION AND GRADIENT CALLS THAT CONMIN WILL
C                     BE ALLOWED TO MAKE.
C             W       W IS A VECTOR OF WORKING STORAGE.IF NMETH=Ø,
C                     W MUST BE DIMENSIONED 5*N+2. IF NMETH=1,
C                     W MUST BE DIMENSIONED N*(N+7)/2. IN BOTH CASES,
C                     W MUST BE DOUBLE PRECISIONED.
C             IOUT    IOUT IS A USER  SUPPLIED OUTPUT PARAMETER.
C                     IF IOUT = Ø, THERE IS NO PRINTED OUTPUT FROM
C                     CONMIN. IF IOUT J Ø,THE VALUE OF F AND THE
C                     NORM OF THE GRADIENT SQUARED,AS WELL AS ITER
C                     AND IFUN,ARE WRITTEN EVERY IOUT ITERATIONS.
C             MDIM    MDIM IS THE USER SUPPLIED DIMENSION OF THE
C                     VECTOR W. IF NMETH=Ø,MDIM=5*N+2. IF NMETH=1,
```

```
C                     MDIM=N*(N+7)/2.
C               IDEV  IDEV IS THE USER SUPPLIED NUMBER OF THE OUTPUT
C                     DEVICE ON WHICH OUTPUT IS TO BE WRITTEN WHEN
C                     IOUTJØ.
C               ACC   ACC IS A USER SUPPLIED ESTIMATE OF MACHINE
C                     ACCURACY. A LINEAR SEARCH IS UNSUCCESSFULLY
C                     TERMINATED WHEN THE NORM OF THE STEP SIZE
C                     BECOMES SMALLER THAN ACC. IN PRACTICE,
C                     ACC=1Ø.D-2Ø HAS PROVED SATISFACTORY. ACC IS
C                     DOUBLE PRECISIONED.
C               NMETH NMETH IS THE USER SUPPLIED VARIABLE WHICH
C                     CHOOSES THE METHOD OF OPTIMIZATION. IF
C                     NMETH=Ø,A CONJUGATE GRADIENT METHOD IS
C                     USED. IF NMETH=1, THE BFGS METHOD IS USED.
C
C REMARKS:      IN ADDITION TO THE SPECIFIED VALUES IN THE ABOVE
C               ARGUMENT LIST, THE USER MUST SUPPLY A SUBROUTINE
C               CALCFG WHICH CALCULATES THE FUNCTION AND GRADIENT AT
C               X AND PLACES THEM IN F AND G(1),...,G(N) RESPECTIVELY.
C               THE SUBROUTINE MUST HAVE THE FORM:
C                     SUBROUTINE CALCFG(N,X,F,G)
C                     DOUBLE PRECISION X(N),G(N),F
C
C               AN EXAMPLE SUBROUTINE FOR THE ROSENBROCK FUNCTION IS:
C
C                     SUBROUTINE CALCFG(N,X,F,G)
C                     DOUBLE PRECISION X(N),G(N),F,T1,T2
C                     T1=X(2)-X(1)*X(1)
C                     T2=1.Ø-X(1)
C                     F=1ØØ.Ø*T1*T1+T2*T2
C                     G(1)=-4ØØ.Ø*T1*X(1)-2.Ø*T2
C                     G(2)=2ØØ.Ø*T1
C                     RETURN
C                     END
```

ALGORITHM 501
Fortran Translation of Algorithm 409, Discrete Chebychev Curve Fit [E2]

JOSEPH C. SIMPSON

Miami University

Key Words and Phrases: approximation, polynomial approximation, exchange algorithm, Chebychev approximation, Remez algorithm, minimax polynomial approximation
CR Categories: 5.11, 5.13
Language: Fortran

DESCRIPTION

Subprograms *APPROX* and *EXCH* fit N data pairs stored with independent variable values in array X and dependent variable values in array Y to a minimax polynomial of degree M with parity specified by variable K (0, mixed parity; 1, odd parity; 2, even parity). The polynomial coefficients are returned in ascending order in array A. The magnitude of floating point variable *EPSH* must be significant when compared with the one on the machine used. The algorithm will fail only if *ABS*(*EPSH*) is greater than *HMAX*, the maximum approximation error. Success or failure of a run is indicated in the fixed point variable *EQUAL*. All other arguments are identical with the variables in the original algorithm and must conform to the constraints mentioned there. The variable *NTAPE*, an internal constant in subroutine *APPROX*, specifies the output unit number to which error messages will be directed. To insure the widest possible utility, a primitive level of Fortran was used. By modifying the code, users with access to powerful compilers can reduce both the amount of code and execution time. In particular, the variable *ITEMP* can be eliminated. The algorithm has been used on Sigma 6, Sigma 9, and IBM 360RAX computers.

ACKNOWLEDGMENT

The author is indebted to Herbert Schmitt for suggestions which improved the subprogram.

REFERENCES

SCHMITT, H. Discrete Chebychev curve fit, Algorithm 409. *Comm. ACM 14*, 5 (May 1971), 355–356.

ALGORITHM

```
      SUBROUTINE APPROX(M, N, K, X, Y, EPSH, REF, MAXIT, HMAX, H,      APP   10
     * A, EQUAL)                                                        APP   20
C THIS SUBROUTINE, IN CONJUNCTION WITH SUBROUTINE EXCH,                 APP   30
```

Received 18 March 1974.

During the time this algorithm was developed, the author was supported by the National Science Foundation and the Petroleum Research Fund. Computer facilities were provided by Marquette University.
Author's address: Academic Computer Service, Miami University, Oxford, OH 45056.

```
C COMPUTES THE MINIMAX (CHEBYCHEV) POLYNOMIAL WHICH FITS              APP  40
C THE DATA IN X AND Y.                                                APP  50
C THIS SUBPROGRAM IS A TRANSLATION FROM ALGOL OF                      APP  60
C H. SCHMITT*S ALGORITHM NUMBER 409 IN CACM, V14,                     APP  70
C PP. 355-356(1971).                                                  APP  80
C THE PRIMARY REFERENCE IS STIEFEL, E. L.  NUMERICAL                  APP  90
C METHODS OF CHEBYCHEV APPROXIMATION, IN *ON NUMERICAL                APP 100
C APPROXIMATION*, R. LANGER, (EDITOR), UNIVERSITY OF                  APP 110
C WISCONSIN PRESS, 1958, PP.217-232.                                  APP 120
C TRANSLATION BY JOSEPH C. SIMPSON, MARQUETTE UNIVERSITY.             APP 130
C THE EXCHANGE ALGORITHM IS A FINITE ITERITIVE PROCESS                APP 140
C REQUIRING, AT MOST, BICO(N,P) ITERATIONS, WHERE P =                 APP 150
C NUMBER OF POINTS FIT IN ONE ITERATION (SEE EXCH FOR THE             APP 160
C VALUE OF P).  SINCE BICO(N,P) CAN BE VERY LARGE, IT IS              APP 170
C POSSIBLE THAT THE ROUTINE WILL NOT CONVERGE WITHIN MAXIT            APP 180
C ITERATIONS.  THE OTHER POSSIBILITY OF FAILURE OCCURS                APP 190
C WHEN INSUFFICIENT FLOATING POINT PRECISION IS AVAILABLE             APP 200
C FOR THE INPUT DATA CHOSEN.                                          APP 210
C              DESCRIPTION OF INPUT-OUTPUT VARIABLES.                 APP 220
C M = DEGREE OF FITTING POLYNOMIAL.                                   APP 230
C N = NUMBER OF DATA POINTS STORED IN X AND Y.                        APP 240
C X = INDEPENDENT VARIABLE VALUES STORED IN ASCENDING ORDER.          APP 250
C Y = DEPENDENT VARIABLE VALUES CORRESPONDING TO ABOVE DATA.          APP 260
C K CONTROLS THE CHARACTER OF THE POLYNOMIAL.                         APP 270
C K = 0 IMPLIES MIXED PARITY POLYNOMIAL.  X(1) MAY BE ANY             APP 280
C FLOATING POINT VALUE.                                               APP 290
C K = 1 IMPLIES ODD POLYNOMIAL.  X(1) MUST BE .GT.0.                  APP 300
C K = 2 IMPLIES EVEN POLYNOMIAL.  X(1) MUST BE .GE.0.                 APP 310
C EPSH.  ABS(EPSH) = TOLERENCE FOR LEVELING.  IF EPSH IS              APP 320
C NEGATIVE, REF CONTAINS THE INITIAL POINT SET.  ABS(EPSH)            APP 330
C SHOULD BE SIGNIFICANT WHEN COMPARED WITH ONE.  A USEFUL             APP 340
C VALUE FOR 24 BIT MANTISSA IS EPSH=2.E-7.                            APP 350
C REF = AN ARRAY OF INITIAL POINTS IF EPSH.LT.0.  OTHERWISE           APP 360
C IT IS NOT AN INPUT ARRAY.  UPON RETURN IT CONTAINS THE              APP 370
C MAXIMUM DEVIATION POINTS.                                           APP 380
C MAXIT = UPPER LIMIT FOR NUMBER OF EXCHANGE STEPS.                   APP 390
C HMAX CONTAINS THE MAXIMUM DEVIATION(OUTPUT).                        APP 400
C H CONTAINS THE APPROXIMATION ERRORS(OUTPUT).                        APP 410
C A CONTAINS THE POLYNOMIAL COEFFICIENTS IN ORDER OF                  APP 420
C INCREASING POWERS(OUTPUT). IF MAXIT IS EXCEEDED, REF                APP 430
C CONTAINS A NEW REFERENCE SET.                                       APP 440
C DIMENSION REF(REFMAX),DAA(M+1),D(M+2),Z(M+4),A(M+1),                APP 450
C AA(M+1),C(M+2),X(N),Y(N),H(N)                                       APP 460
C REFMAX(K=0)=M+2. REFMAX(K=1)=INT((M+3)/2).                          APP 470
C REFMAX(K=2)=2+M/2.                                                  APP 480
C INPUT VARIABLES ALTERED DURING EXECUTION CAN INCLUDE M,             APP 490
C EPSH,REF.                                                           APP 500
C EQUAL RECORDS THE SUCCESS OR FAILURE OF THE ALGORITHM.              APP 510
C EQUAL=1  SUCCESFUL.                                                 APP 520
C EQUAL=0  CONVERGENCE OF EXCHANGE PROCESS NOT ACHEIVED.              APP 530
C EQUAL=-1  INPUT ERROR.                                              APP 540
C EQUAL=-2  ALGORITHM FAILURE.  NOTE.. A MAY CONTAIN THE              APP 550
C            COLLOCATION POLYNOMIAL.                                  APP 560
C          DESCRIPTION OF SOME LOCAL VARIABLES.                       APP 570
C SMALLEST REPRESENTABLE NUMBER.  IT IS USED ONLY WHEN                APP 580
C COLLOCATION IS ENCOUNTERED.                                         APP 590
C THE FOLLOWING VARIABLES ARE DECLARED INTEGER, BUT ARE USED          APP 600
C AS LOGICAL VARIABLES ONLY,  .TRUE.=1  AND  .FALSE.=0                APP 610
C K0,K1,B1,B2.                                                        APP 620
C Z(I+1) CONTAINS THE ITH POINT OF THE CURRENT REFERENCE              APP 630
C SET.  DAA(I+1) CONTAINS THE CORRECTION TO THE COEFFICIENT           APP 640
C OF X**(I).                                                          APP 650
C TO OBTAIN A DOUBLE PRECISION VERSION, CHANGE THE                    APP 660
C REAL DECLARATION TO DOUBLE PRECISION, ABS TO DABS, COS              APP 670
C TO DCOS, AND FLOAT TO DFLOAT.                                       APP 680
C THIS DIMENSION STATEMENT ALLOWS 50 POINTS AND A MIXED               APP 690
C ORDER POLYNOMIAL OF DEGREE 30.                                      APP 700
      DIMENSION XX(50), AA(31), DAA(31), D(32), Z(34), C(32)          APP 710
      DIMENSION X(1), Y(1), H(1), REF(1), A(1)                        APP 720
      DOUBLE PRECISION SD, QD, C, T, D, DT, AA, DAA, XX, MAX          APP 730
      REAL EPSH, HMAX, Y, H, A, PI, Q, S, X1, XA, XE, COS             APP 740
      INTEGER N, M, K, MAXIT, REF, I, J, P, Q1, Q2, R, K0, K1, B1,    APP 750
     * B2, Z, ITEMP, MOLD, HIGH, JJ, II, LOW, EQUAL, NTAPE, B3         APP 760
C MACHINE DEPENDENT CONSTANTS.                                        APP 770
C NTAPE IS THE PRINTER UNIT NUMBER.                                   APP 780
      NTAPE = 6                                                       APP 790
      PI = 3.1415926535                                               APP 800
      MOLD = M                                                        APP 810
C ZERO THE COEFFICIENT ARRAY.                                         APP 820
      IF (K-1) 10, 20, 30                                             APP 830
   10 K0 = 1                                                          APP 840
      K1 = 0                                                          APP 850
      Q1 = 0                                                          APP 860
      Q2 = 1                                                          APP 870
      GO TO 50                                                        APP 880
   20 K0 = 0                                                          APP 890
      K1 = 1                                                          APP 900
      Q1 = 1                                                          APP 910
      Q2 = 2                                                          APP 920
      GO TO 40                                                        APP 930
   30 K0 = 0                                                          APP 940
```

```
      K1 = 0                                                          APP  950
      Q1 = 0                                                          APP  960
      Q2 = 2                                                          APP  970
C ADJUST M ACCORDING TO THE VALUE OF K                                APP  980
   40 M = (M-Q1)/2                                                    APP  990
   50 P = M + 2                                                       APP 1000
C SCREEN INPUT PARAMETERS.                                            APP 1010
      IF (P-N) 60, 60, 780                                            APP 1020
   60 IF (M) 780, 70, 70                                              APP 1030
   70 IF (K-1) 100, 80, 90                                            APP 1040
   80 IF (X(1)) 790, 790, 100                                         APP 1050
   90 IF (X(1)) 790, 100, 100                                         APP 1060
  100 DO 120 I=2,N                                                    APP 1070
        ITEMP = I - 1                                                 APP 1080
        IF (X(I)-X(ITEMP)) 800, 110, 110                              APP 1090
  110   CONTINUE                                                      APP 1100
  120 CONTINUE                                                        APP 1110
      ITEMP = MOLD + 1                                                APP 1120
      DO 130 I=1,ITEMP                                                APP 1130
        A(I) = 0.                                                     APP 1140
  130 CONTINUE                                                        APP 1150
      Z(1) = 0                                                        APP 1160
      ITEMP = P + 2                                                   APP 1170
      Z(ITEMP) = N + 1                                                APP 1180
      IF (EPSH) 140, 170, 170                                         APP 1190
C IF EPSH.GT.0, BRANCH TO Z SETUP SECTION.                            APP 1200
C IF EPSH IS NEGATIVE, TEST REF AND LOAD IT INTO Z.                   APP 1210
  140 EPSH = -EPSH                                                    APP 1220
      J = 0                                                           APP 1230
      DO 160 I=1,P                                                    APP 1240
        ITEMP = I + 1                                                 APP 1250
        R = REF(I)                                                    APP 1260
        Z(ITEMP) = R                                                  APP 1270
C BRANCH TO ERROR EXIT UNLESS REF IS MONOTONICALLY                    APP 1280
C INCREASING.                                                         APP 1290
        IF (J-R) 150, 810, 810                                        APP 1300
  150   J = R                                                         APP 1310
  160 CONTINUE                                                        APP 1320
      IF (J-N) 300, 300, 810                                          APP 1330
C BRANCH AROUND Z SETUP SECTION.                                      APP 1340
C THIS SECTION LOADS Z WITH THE POINTS CLOSEST TO THE                 APP 1350
C CHEBYCHEV ABSCISSAS.                                                APP 1360
  170 X1 = X(1)                                                       APP 1370
      XE = X(N)                                                       APP 1380
      IF (K0) 190, 190, 180                                           APP 1390
C IF THE POLYNOMIAL IS NOT MIXED, I.E.HAS A DEFINITE PARITY,          APP 1400
C BRANCH TO 20.                                                       APP 1410
  180 XA = XE + X1                                                    APP 1420
      XE = XE - X1                                                    APP 1430
      Q = PI/FLOAT(M+1)                                               APP 1440
      GO TO 200                                                       APP 1450
  190 XA = 0.                                                         APP 1460
      XE = XE + XE                                                    APP 1470
      ITEMP = 2*(M+1) + Q1                                            APP 1480
      Q = PI/FLOAT(ITEMP)                                             APP 1490
C CALCULATE THE JTH CHEBYCHEV ABSCISSA AND LOAD Z(J+1) WITH           APP 1500
C THE APPROPRIATE INDEX FROM THE DATA ABSCISSAS.                      APP 1510
  200 DO 270 JJ=1,P                                                   APP 1520
        J = P + 1 - JJ                                                APP 1530
        X1 = XA + XE*(COS(Q*FLOAT(P-J)))                              APP 1540
        ITEMP = J + 2                                                 APP 1550
        R = Z(ITEMP)                                                  APP 1560
        HIGH = R - 1                                                  APP 1570
        IF (HIGH-2) 230, 210, 210                                     APP 1580
  210   DO 220 II=2,HIGH                                              APP 1590
          I = HIGH + 2 - II                                           APP 1600
          ITEMP = I - 1                                               APP 1610
          IF (X(I)+X(ITEMP)-X1) 240, 240, 220                         APP 1620
C WHEN THE CHEBYCHEV ABSCISSA IS BRACKETED BY TWO INPUT               APP 1630
C ABSCISSAS, BRANCH TO 240.                                           APP 1640
  220   CONTINUE                                                      APP 1650
  230   I = 1                                                         APP 1660
  240   ITEMP = J + 1                                                 APP 1670
        IF (I-R) 250, 260, 260                                        APP 1680
  250   Z(ITEMP) = I                                                  APP 1690
        GO TO 270                                                     APP 1700
  260   Z(ITEMP) = R - 1                                              APP 1710
  270 CONTINUE                                                        APP 1720
C IF THE LOWER CHEBYCHEV ABSCISSAS ARE LESS THAN X(1), LOAD           APP 1730
C THE LOWER ELEMENTS OF Z WITH THE LOWEST POINTS.                     APP 1740
      J = 0                                                           APP 1750
  280 J = J + 1                                                       APP 1760
      ITEMP = J + 1                                                   APP 1770
      IF (Z(ITEMP)-J) 290, 300, 300                                   APP 1780
  290 Z(ITEMP) = J                                                    APP 1790
      GO TO 280                                                       APP 1800
C M1 ENTRY.  INITIALIZE VARIABLES TO PREPARE FOR EXCHANGE             APP 1810
C ITERATION.                                                          APP 1820
  300 ITEMP = M + 1                                                   APP 1830
C ZERO THE AA ARRAY.                                                  APP 1840
      DO 310 I=1,ITEMP                                                APP 1850
```

```
      AA(I) = 0.                                                    APP 1860
  310 CONTINUE                                                      APP 1870
C LOAD H WITH THE ORDINATES AND XX(I) WITH THE ABSCISSAS IF         APP 1880
C THE POLYNOMIAL IS MIXED.  IF THE POLYNOMIAL IS EVEN OR ODD        APP 1890
C LOAD XX WITH THE SQUARES OF THE ABSCISSAS.                        APP 1900
      DO 340 I=1,N                                                  APP 1910
        H(I) = Y(I)                                                 APP 1920
        Q = X(I)                                                    APP 1930
        IF (K0) 320, 320, 330                                       APP 1940
  320   XX(I) = Q*Q                                                 APP 1950
        GO TO 340                                                   APP 1960
  330   XX(I) = Q                                                   APP 1970
  340 CONTINUE                                                      APP 1980
      B1 = 0                                                        APP 1990
      B2 = 0                                                        APP 2000
      B3 = 0                                                        APP 2010
      R = -1                                                        APP 2020
      T = 0.                                                        APP 2030
C ITERATION ENTRY.  R IS THE ITERATION INDEX.                       APP 2040
  350 R = R + 1                                                     APP 2050
      S = 1.                                                        APP 2060
C COMPUTATION OF DIVIDED DIFFERENCES SCHEME.                        APP 2070
      IF (K1) 380, 380, 360                                         APP 2080
C IF THE POLYNOMIAL IS ODD, BRANCH TO 380.                          APP 2090
  360 DO 370 I=1,P                                                  APP 2100
        S = -S                                                      APP 2110
        ITEMP = I + 1                                               APP 2120
        J = Z(ITEMP)                                                APP 2130
        Q = X(J)                                                    APP 2140
        C(I) = (H(J)+S*T)/Q                                         APP 2150
        D(I) = S/Q                                                  APP 2160
  370 CONTINUE                                                      APP 2170
      GO TO 400                                                     APP 2180
  380 DO 390 I=1,P                                                  APP 2190
        S = -S                                                      APP 2200
        ITEMP = I + 1                                               APP 2210
        ITEMP = Z(ITEMP)                                            APP 2220
        C(I) = H(ITEMP) + S*T                                       APP 2230
        D(I) = S                                                    APP 2240
  390 CONTINUE                                                      APP 2250
  400 CONTINUE                                                      APP 2260
      DO 420 I=2,P                                                  APP 2270
        DO 410 JJ=I,P                                               APP 2280
          J = P + I - JJ                                            APP 2290
          ITEMP = J + 1                                             APP 2300
          ITEMP = Z(ITEMP)                                          APP 2310
          QD = XX(ITEMP)                                            APP 2320
          ITEMP = 2 + J - I                                         APP 2330
          ITEMP = Z(ITEMP)                                          APP 2340
          QD = QD - XX(ITEMP)                                       APP 2350
          ITEMP = J - 1                                             APP 2360
          C(J) = (C(J)-C(ITEMP))/QD                                 APP 2370
          D(J) = (D(J)-D(ITEMP))/QD                                 APP 2380
  410   CONTINUE                                                    APP 2390
  420 CONTINUE                                                      APP 2400
      DT = -C(P)/D(P)                                               APP 2410
      T = T + DT                                                    APP 2420
C COMPUTATION OF POLYNOMIAL COEFFICIENTS.                           APP 2430
      HIGH = M + 1                                                  APP 2440
      DO 450 II=1,HIGH                                              APP 2450
        I = HIGH - II                                               APP 2460
        ITEMP = I + 1                                               APP 2470
        DAA(ITEMP) = C(ITEMP) + DT*D(ITEMP)                         APP 2480
        ITEMP = I + 2                                               APP 2490
        ITEMP = Z(ITEMP)                                            APP 2500
        QD = XX(ITEMP)                                              APP 2510
        LOW = I + 1                                                 APP 2520
        IF (M-LOW) 450, 430, 430                                    APP 2530
  430   DO 440 J=LOW,M                                              APP 2540
          JJ = J + 1                                                APP 2550
          DAA(J) = DAA(J) - QD*DAA(JJ)                              APP 2560
  440   CONTINUE                                                    APP 2570
  450 CONTINUE                                                      APP 2580
      DO 460 I=1,HIGH                                               APP 2590
        AA(I) = AA(I) + DAA(I)                                      APP 2600
  460 CONTINUE                                                      APP 2610
C EVALUATION OF THE POLYNOMIAL TO GET THE APPROXIMATION             APP 2620
C ERRORS.                                                           APP 2630
      MAX = 0.                                                      APP 2640
      DO 540 I=1,N                                                  APP 2650
        SD = AA(HIGH)                                               APP 2660
        QD = XX(I)                                                  APP 2670
        IF (-M) 470, 490, 490                                       APP 2680
  470   DO 480 JJ=1,M                                               APP 2690
          J = M - JJ + 1                                            APP 2700
          SD = SD*QD + AA(J)                                        APP 2710
  480   CONTINUE                                                    APP 2720
  490   CONTINUE                                                    APP 2730
        IF (K1) 510, 510, 500                                       APP 2740
C IF THE POLYNOMIAL IS  ODD MULTIPLY SD BY X(I).                    APP 2750
```

```
  500    SD = SD*X(I)                                                  APP 2760
  510    QD = Y(I) - SD                                                APP 2770
         H(I) = QD                                                     APP 2780
         IF (DABS(QD)-MAX) 530, 530, 520                               APP 2790
C LOAD MAX WITH THE LARGEST MAGNITUDE OF THE APPROXIMATION             APP 2800
C ARRAY.                                                               APP 2810
  520    MAX = DABS(QD)                                                APP 2820
  530    CONTINUE                                                      APP 2830
  540 CONTINUE                                                         APP 2840
C TEST FOR ALTERNATING SIGNS.                                          APP 2850
       ITEMP = Z(2)                                                    APP 2860
       IF (H(ITEMP)) 550, 560, 600                                     APP 2870
  550 J = 1                                                            APP 2880
       GO TO 610                                                       APP 2890
  560 J = 0                                                            APP 2900
C THIS REPRESENTS A CASE WHERE THE POLYNOMIAL EXACTLY                  APP 2910
C PREDICTS A DATA POINT.                                               APP 2920
       WRITE (NTAPE,99998)                                             APP 2930
       IF (B3) 570, 570, 690                                           APP 2940
  570 B3 = 1                                                           APP 2950
       IF (EPSH-MAX) 580, 710, 710                                     APP 2960
  580 WRITE (NTAPE,99999)                                              APP 2970
       LOW = (N+1)/2 - (P+1)/2 + 1                                     APP 2980
       HIGH = LOW + P                                                  APP 2990
       DO 590 I=LOW,HIGH                                               APP 3000
         Z(I) = I - 1                                                  APP 3010
  590 CONTINUE                                                         APP 3020
       GO TO 350                                                       APP 3030
  600 J = -1                                                           APP 3040
  610 CONTINUE                                                         APP 3050
       DO 670 I=2,P                                                    APP 3060
         ITEMP = I + 1                                                 APP 3070
         ITEMP = Z(ITEMP)                                              APP 3080
C IF(H(ITEMP)) 339,340,341                                             APP 3090
         IF (H(ITEMP)) 620, 560, 630                                   APP 3100
  620    JJ = -1                                                       APP 3110
         GO TO 640                                                     APP 3120
  630    JJ = 1                                                        APP 3130
  640    IF (J-JJ) 650, 660, 650                                       APP 3140
C ERROR ENTRY.  INSUFFICIENT ACCURACY FOR CALCULATION.                 APP 3150
  650    B1 = 1                                                        APP 3160
         GO TO 720                                                     APP 3170
  660    J = -J                                                        APP 3180
  670 CONTINUE                                                         APP 3190
C SEARCH FOR ANOTHER REFERENCE.                                        APP 3200
       CALL EXCH(N, P, H, EPSH, Z, EQUAL)                              APP 3210
       IF (EQUAL) 680, 680, 720                                        APP 3220
  680 IF (R-MAXIT) 350, 700, 700                                       APP 3230
  690 B3 = -1                                                          APP 3240
       GO TO 720                                                       APP 3250
  700 B2 = 1                                                           APP 3260
       GO TO 720                                                       APP 3270
  710 WRITE (NTAPE,99988) MAX                                          APP 3280
C END OF ITERATION LOOP.                                               APP 3290
C M2 ENTRY.  LOAD OUTPUT VARIABLES AND RETURN.                         APP 3300
  720 HIGH = M + 1                                                     APP 3310
C LOAD THE COEFFICIENTS INTO THE A ARRAY.                              APP 3320
       DO 730 I=1,HIGH                                                 APP 3330
         ITEMP = Q1 + Q2*(I-1) + 1                                     APP 3340
         A(ITEMP) = AA(I)                                              APP 3350
  730 CONTINUE                                                         APP 3360
C LOAD REF WITH THE FINAL REFERENCE POINTS.                            APP 3370
       DO 740 I=1,P                                                    APP 3380
         ITEMP = I + 1                                                 APP 3390
         REF(I) = Z(ITEMP)                                             APP 3400
  740 CONTINUE                                                         APP 3410
       HMAX = MAX                                                      APP 3420
       MAXIT = R                                                       APP 3430
       IF (B3) 840, 750, 750                                           APP 3440
  750 IF (B1) 760, 760, 820                                            APP 3450
  760 IF (B2) 770, 770, 830                                            APP 3460
  770 M = MOLD                                                         APP 3470
C NORMAL EXIT                                                          APP 3480
       RETURN                                                          APP 3490
C ERROR EXITS.                                                         APP 3500
  780 WRITE (NTAPE,99990)                                              APP 3510
       WRITE (NTAPE,99996) N, MOLD, K                                  APP 3520
       EQUAL = -1                                                      APP 3530
       GO TO 770                                                       APP 3540
  790 WRITE (NTAPE,99990)                                              APP 3550
       WRITE (NTAPE,99995) K, X(1)                                     APP 3560
       EQUAL = -1                                                      APP 3570
       GO TO 770                                                       APP 3580
  800 WRITE (NTAPE,99990)                                              APP 3590
       WRITE (NTAPE,99994) (X(I),I=1,N)                                APP 3600
       EQUAL = -1                                                      APP 3610
       GO TO 770                                                       APP 3620
  810 WRITE (NTAPE,99990)                                              APP 3630
       WRITE (NTAPE,99993) (REF(I),I=1,N)                              APP 3640
       EQUAL = -1                                                      APP 3650
       GO TO 770                                                       APP 3660
```

```
  820 WRITE (NTAPE,99997)                                                APP 3670
      WRITE (NTAPE,99992) HMAX                                           APP 3680
      EQUAL = -2                                                         APP 3690
      GO TO 770                                                          APP 3700
  830 WRITE (NTAPE,99997)                                                APP 3710
      ITEMP = M + 2                                                      APP 3720
      WRITE (NTAPE,99991) MAXIT, (REF(I),I=1,ITEMP)                      APP 3730
      EQUAL = 0                                                          APP 3740
      GO TO 770                                                          APP 3750
  840 WRITE (NTAPE,99997)                                                APP 3760
      WRITE (NTAPE,99989)                                                APP 3770
      EQUAL = -2                                                         APP 3780
      GO TO 770                                                          APP 3790
99999 FORMAT (1H+, 26X, 35HONE MORE ATTEMPT WILL BE MADE USING,          APP 3800
     * 19H THE MIDDLE POINTS.)                                           APP 3810
99998 FORMAT (25H COLLOCATION HAS OCCURRED.)                             APP 3820
99997 FORMAT (26H ALGORITHM APPROX FAILURE.)                             APP 3830
99996 FORMAT (10X, 28HINSUFFICIENT INFORMATION. N=, I3, 3H M=, I3,       APP 3840
     * 3H K=, I3)                                                        APP 3850
99995 FORMAT (10X, 37HPOLYNOMIAL PARITY - ABSCISSA CONFLICT.,            APP 3860
     * 6H  K = , I3, 7H  X(1)=, F20.10)                                  APP 3870
99994 FORMAT (10X, 40HABSCISSAS OUT OF ORDER.  X ARRAY EQUALS            APP 3880
     * /(7E17.7))                                                        APP 3890
99993 FORMAT (10X, 40HINITIAL POINT ARRAY NOT MONOTONIC INCREA,          APP 3900
     * 23HSING.  REF ARRAY EQUALS/(13I9))                                APP 3910
99992 FORMAT (1X, 39HAPPROXIMATION ERRORS AT POINTS OF REFER,            APP 3920
     * 30HENCE DO NOT ALTERNATE IN SIGN./23H SUSPECT INSUFFICIENT W,     APP 3930
     * 11HORD LENGTH., 16H MAXIMUM ERROR= , E15.7/14H IF POLYNOMIAL,     APP 3940
     * 34H WAS ASSUMED TO BE OF MIXED PARITY, 18H, CHECK FOR EVEN O,     APP 3950
     * 13HR ODD PARITY.)                                                 APP 3960
99991 FORMAT (31H MAXIMUM NUMBER OF ITERATIONS, , I6, 10H THE REFER,     APP 3970
     * 23HENCE ARRAY OBTAINED IS /(13I9))                               APP 3980
99990 FORMAT (31H INPUT ERROR SUBPROGRAM APPROX.)                        APP 3990
99989 FORMAT (1H+, 26X, 18HAPPROX TERMINATES.)                           APP 4000
99988 FORMAT (1H+, 26X, 6H MAX= , E15.7, 23H .LT.HMIN.  NORMAL EXIT,     APP 4010
     * 1H.)                                                              APP 4020
      END                                                                APP 4030


      SUBROUTINE EXCH(N, P, H, EPSH, Z, EQUAL)                           EXC   10
C DIMENSION H(N),Z(P+2)                                                  EXC   20
C N = NUMBER OF POINTS.                                                  EXC   30
C P = NUMBER OF REFERENCE POINTS.                                        EXC   40
C EPSH = APPROXIMATION ERROR STANDARD.                                   EXC   50
C Z= INPUT   OLD REFERENCE INDICES.                                      EXC   60
C    OUTPUT  NEW REFERENCE INDICES.                                      EXC   70
C Z(1) SHOULD CONTAIN ZERO.                                              EXC   80
C Z(P+2) SHOULD CONTAIN (N+1).                                           EXC   90
C Z(I+1) CONTAINS THE ITH REFERENCE POINT INDEX.                         EXC  100
C H = ARRAY OF APPROXIMATION ERRORS.                                     EXC  110
C EQUAL = 0 IMPLIES NORMAL EXCHANGE.                                     EXC  120
C EQUAL = 1 IMPLIES OLD AND NEW REFERENCE SETS ARE EQUAL.                EXC  130
C THE APPROXIMATION ERRORS ARE COMPARED RELATIVE TO EPSH.                EXC  140
C          DESCRIPTION OF SOME LOCAL VARIABLES.                          EXC  150
C HZ1 CONTAINS THE LOW POINT APPROXIMATION ERROR.                        EXC  160
C HZP CONTAINS THE HIGH POINT APPROXIMATION ERROR.                       EXC  170
C MAXR CONTAINS THE LARGEST APPROXIMATION ERROR ABOVE THE                EXC  180
C REFERENCE POINT SET. INDR IS THE POINT INDEX FOR MAXR.                 EXC  190
C MAXL CONTAINS THE LARGEST APPROXIMATION ERROR BELOW THE                EXC  200
C REFERENCE POINT SET. INDL IS THE POINT INDEX FOR MAXL.                 EXC  210
C ITEMP IS A FIXED POINT VARIABLE USED FOR TEMPORARY STORAGE             EXC  220
C CODING FOR HIGHER LEVEL FORTRAN COULD ELIMINATE ITEMP.                 EXC  230
C TO OBTAIN A DOUBLE PRECISION VERSION, CHANGE THE REAL                  EXC  240
C DECLARATION TO DOUBLE PRECISION AND ABS TO DABS.                       EXC  250
      DIMENSION Z(1), H(1)                                               EXC  260
      INTEGER Z, N, P, I, J, L, INDEX, INDL, INDR, ZE, ITEMP, LOW,       EXC  270
     * HIGH, EQUAL, II                                                   EXC  280
      REAL EPSH, H, HZ1, HZP, MAX, MAXL, MAXR, SIG                       EXC  290
      EQUAL = 0                                                          EXC  300
      L = 0                                                              EXC  310
      ITEMP = Z(2)                                                       EXC  320
C ARBITRARILY CHOSEN EQUAL TO THE SIGN OF THE INPUT POINT.               EXC  330
C THIS WILL BE ADJUSTED LATER IF NECESSARY.                              EXC  340
      IF (H(ITEMP)) 10, 10, 20                                           EXC  350
   10 SIG = 1.                                                           EXC  360
      GO TO 30                                                           EXC  370
   20 SIG = -1.                                                          EXC  380
   30 DO 90 I=1,P                                                        EXC  390
C DO 90 PRESCANS Z TO INSURE IT IS A PROPER CHOICE, I. E.                EXC  400
C RESETS Z IF NECESSARY SO THAT MAXIMUM ERROR POINTS ARE                 EXC  410
C CHOSEN, GIVEN THE SIGN CONVENTION MENTIONED ABOVE.                     EXC  420
C IN ORDER TO WORK PROPERLY, THIS SECTION REQUIRES Z(1)=0                EXC  430
C AND Z(P+2)=N+1.                                                        EXC  440
        MAX = 0.                                                         EXC  450
        SIG = -SIG                                                       EXC  460
        ITEMP = I + 2                                                    EXC  470
        ZE = Z(ITEMP) - 1                                                EXC  480
        LOW = Z(I) + 1                                                   EXC  490
C SCAN THE OPEN POINT INTERVAL CONTAINING ONLY THE ITH                   EXC  500
C INITIAL REFERENCE POINT. IN THE INTERVAL PICK THE                      EXC  510
```

```
C POINT WITH LARGEST MAGNITUDE AND CORRECT SIGN.  MOST OF                EXC  520
C THE SORTING OCCURS IN THIS SECTION.  SIG CONTAINS THE                  EXC  530
C SIGN ASSUMED FOR H(I).                                                 EXC  540
      DO 60 J=LOW,ZE                                                     EXC  550
        IF (SIG*(H(J)-MAX)) 50, 50, 40                                   EXC  560
 40     MAX = H(J)                                                       EXC  570
        INDEX = J                                                        EXC  580
 50     CONTINUE                                                         EXC  590
 60   CONTINUE                                                           EXC  600
      ITEMP = I + 1                                                      EXC  610
      ITEMP = Z(ITEMP)                                                   EXC  620
      MAXL = ABS(MAX)                                                    EXC  630
      IF (ABS(MAX-H(ITEMP))/MAXL-EPSH) 80, 80, 70                        EXC  640
C IF THE MAX ERROR IS SIGNIFICANTLY GREATER THAN THE INPUT               EXC  650
C POINT SWITCH TO THIS POINT.                                            EXC  660
 70   ITEMP = I + 1                                                      EXC  670
      Z(ITEMP) = INDEX                                                   EXC  680
      L = 1                                                              EXC  690
 80   CONTINUE                                                           EXC  700
 90 CONTINUE                                                             EXC  710
    MAXL = 0.                                                            EXC  720
    MAXR = 0.                                                            EXC  730
    ITEMP = P + 1                                                        EXC  740
    LOW = Z(ITEMP) + 1                                                   EXC  750
    IF (LOW-N) 100, 100, 140                                             EXC  760
100 CONTINUE                                                             EXC  770
C FIND THE ERROR WITH LARGEST ABSOLUTE VALUE AND PROPER SIGN             EXC  780
C FROM AMONG THE POINTS ABOVE THE LAST REFERENCE POINT.                  EXC  790
C THIS SECTION IS NECESSARY BECAUSE THE SET OF POINTS CHOSEN             EXC  800
C MAY BEGIN WITH THE WRONG SIGN ALTERNATION.                             EXC  810
      DO 130 J=LOW,N                                                     EXC  820
        IF (SIG*(MAXR-H(J))) 120, 120, 110                               EXC  830
110   MAXR = H(J)                                                        EXC  840
      INDR = J                                                           EXC  850
120   CONTINUE                                                           EXC  860
130 CONTINUE                                                             EXC  870
140 CONTINUE                                                             EXC  880
C FIND THE ERROR WITH LARGEST ABSOLUTE VALUE AND PROPER SIGN             EXC  890
C FROM AMONG THE POINTS BELOW THE FIRST REFERENCE POINT.                 EXC  900
C THIS SECTION IS NECESSARY BECAUSE THE SET OF POINTS CHOSEN             EXC  910
C MAY BEGIN WITH THE WRONG SIGN ALTERNATION.                             EXC  920
      ITEMP = Z(2)                                                       EXC  930
      HZ1 = H(ITEMP)                                                     EXC  940
      HIGH = ITEMP - 1                                                   EXC  950
      IF (HIGH) 230, 230, 150                                            EXC  960
150 CONTINUE                                                             EXC  970
      IF (HZ1) 160, 170, 180                                             EXC  980
160 SIG = -1.                                                            EXC  990
      GO TO 190                                                          EXC 1000
170 SIG = 0.                                                             EXC 1010
      GO TO 190                                                          EXC 1020
180 SIG = 1.                                                             EXC 1030
190 CONTINUE                                                             EXC 1040
      DO 220 J=1,HIGH                                                    EXC 1050
        IF (SIG*(MAXL-H(J))) 210, 210, 200                               EXC 1060
200   MAXL = H(J)                                                        EXC 1070
      INDL = J                                                           EXC 1080
210   CONTINUE                                                           EXC 1090
220 CONTINUE                                                             EXC 1100
230 CONTINUE                                                             EXC 1110
      MAXL = ABS(MAXL)                                                   EXC 1120
      MAXR = ABS(MAXR)                                                   EXC 1130
      HZ1 = ABS(HZ1)                                                     EXC 1140
      ITEMP = P + 1                                                      EXC 1150
      ITEMP = Z(ITEMP)                                                   EXC 1160
      HZP = ABS(H(ITEMP))                                                EXC 1170
C MAXL AND MAXR CONTAIN THE MAGNITUDE OF THE SIGNIFICANT                 EXC 1180
C ERRORS OUTSIDE THE REFERENCE POINT SET.  IF EITHER IS                  EXC 1190
C ZERO, THE REFERENCE POINT SET EXTENDS TO THE END POINT                 EXC 1200
C ON THAT SIDE OF THE INTERVAL.                                          EXC 1210
      IF (L) 290, 240, 290                                               EXC 1220
C L=0 IMPLIES THAT THE PRESCAN OF DO 90 DID NOT CHANGE ANY               EXC 1230
C POINTS.IF L=0 AND MAXL AND MAXR ARE NOT SIGNIFICANT WHEN               EXC 1240
C COMPARED WITH UPPER AND LOWER REFERENCE POINT ERRORS,                  EXC 1250
C RESPECTIVELY, USE THE EQUAL EXIT.                                      EXC 1260
240 IF (MAXL) 250, 260, 250                                              EXC 1270
250 IF (EPSH-(MAXL-HZP)/MAXL) 290, 260, 260                              EXC 1280
260 IF (MAXR) 270, 280, 270                                              EXC 1290
270 IF (EPSH-(MAXR-HZ1)/MAXR) 290, 280, 280                              EXC 1300
C EQUAL BRANCH POINT.                                                    EXC 1310
280 EQUAL = 1                                                            EXC 1320
      RETURN                                                             EXC 1330
C ENTER HERE IF ANY CHANGES HAVE BEEN MADE.  THEN TEST TO                EXC 1340
C SEE IF A POINT OUTSIDE THE PRESET POINT SET SHOULD BE                  EXC 1350
C INCLUDED.  IF NOT, RETURN.                                             EXC 1360
290 IF (MAXL) 320, 300, 320                                              EXC 1370
300 IF (MAXR) 320, 310, 320                                              EXC 1380
C RETURN BRANCH (END).                                                   EXC 1390
310 CONTINUE                                                             EXC 1400
      RETURN                                                             EXC 1410
```

```
C IF A POINT OUTSIDE THE PRESENT REFERENCE SET MUST BE                  EXC 1420
C INCLUDED, (I.E.  THE SIGN OF THE FIRST POINT, ASSUMED IN              EXC 1430
C THE DO 90 SECTION IS WRONG) SHIFT TO THE SIDE OF LARGEST              EXC 1440
C RELATIVE ERROR FIRST.                                                 EXC 1450
C CHECK THE OTHER SIDE.                                                 EXC 1460
  320 IF (MAXL-MAXR) 350, 350, 330                                      EXC 1470
  330 IF (MAXL-HZP) 340, 340, 420                                       EXC 1480
  340 IF (MAXR-HZ1) 310, 370, 370                                       EXC 1490
  350 IF (MAXR-HZ1) 360, 360, 370                                       EXC 1500
  360 IF (MAXL-HZP) 310, 420, 420                                       EXC 1510
C SHR ENTRY.                                                            EXC 1520
C THIS SECTION INSERTS A POINT FROM ABOVE THE PRESCAN                   EXC 1530
C POINT SET.                                                            EXC 1540
  370 INDEX = Z(2)                                                      EXC 1550
      DO 380 I=2,P                                                      EXC 1560
C SHIFT POINT SET DOWN, DROPPING THE LOWEST POINT.                      EXC 1570
        ITEMP = I + 1                                                   EXC 1580
        Z(I) = Z(ITEMP)                                                 EXC 1590
  380 CONTINUE                                                          EXC 1600
      ITEMP = P + 1                                                     EXC 1610
C ADD THE NEW HIGH POINT.                                               EXC 1620
      Z(ITEMP) = INDR                                                   EXC 1630
      IF (MAXL) 310, 310, 390                                           EXC 1640
C IF MAXL=0, RETURN.                                                    EXC 1650
C IF MAXL.GT.0  REPLACE REFERENCE POINTS FROM THE LEFT,                 EXC 1660
C STOPPING THE FIRST TIME THE CANDIDATE FOR REPLACEMENT IS              EXC 1670
C GREATER IN MAGNITUDE THAN THE PROSPECTIVE REPLACEE.                   EXC 1680
C ALTERNATION OF SIGNS IS PRESERVED BECAUSE NON-REPLACEMENT             EXC 1690
C IMMEDIATLY TERMINATES THE PROCESS.                                    EXC 1700
  390 DO 410 I=2,P                                                      EXC 1710
        ITEMP = Z(I)                                                    EXC 1720
        IF (ABS(H(INDL))-ABS(H(ITEMP))) 310, 400, 400                   EXC 1730
  400   J = Z(I)                                                        EXC 1740
        Z(I) = INDL                                                     EXC 1750
        INDL = INDEX                                                    EXC 1760
        INDEX = J                                                       EXC 1770
  410 CONTINUE                                                          EXC 1780
      GO TO 310                                                         EXC 1790
C ENTRY SHL.  THIS SECTION INSERTS A POINT FROM BELOW THE               EXC 1800
C PRESCAN POINT SET.                                                    EXC 1810
  420 ITEMP = P + 1                                                     EXC 1820
      INDEX = Z(ITEMP)                                                  EXC 1830
C SHIFT REFERENCE POINT SET UP BY ONE.                                  EXC 1840
      DO 430 II=2,P                                                     EXC 1850
        I = P + 2 - II                                                  EXC 1860
        ITEMP = I + 1                                                   EXC 1870
  430 CONTINUE                                                          EXC 1890
C STORE LOWEST POINT IN Z(2)                                            EXC 1900
      Z(2) = INDL                                                       EXC 1910
      IF (MAXR) 310, 310, 440                                           EXC 1920
C IF MAXR=0, RETURN.                                                    EXC 1930
C IF MAXR.GT.0, START REPLACING REFERENCE POINTS FROM RIGHT,            EXC 1940
C STOPPING THE FIRST TIME THE CANDIDATE FOR REPLACEMENT                 EXC 1950
C IS GREATER IN MAGNITUDE THAN THE PROSPECTIVE REPLACEE.                EXC 1960
  440 DO 460 II=2,P                                                     EXC 1970
        I = P + 2 - II                                                  EXC 1980
        ITEMP = I + 1                                                   EXC 1990
        HIGH = Z(ITEMP)                                                 EXC 2000
        IF (ABS(H(INDR)-ABS(H(HIGH)))) 310, 450, 450                    EXC 2010
  450   J = Z(ITEMP)                                                    EXC 2020
        Z(ITEMP) = INDR                                                 EXC 2030
        INDR = INDEX                                                    EXC 2040
        INDEX = J                                                       EXC 2050
  460 CONTINUE                                                          EXC 2060
      GO TO 310                                                         EXC 2070
      END                                                               EXC 2080
```

ACM Transactions on Mathematical Software, Vol. 4, No. 1, March 1978, Page 95.

REMARK ON ALGORITHM 501

Fortran Translation of Algorithm 409. Discrete Chebychev Curve Fit [E2] [Joseph C. Simpson, *ACM Trans. Math. Software 2*, 1 (March 1976), 95-97]

R. Futrell [Recd 16 February 1977]
Harris Corporation, P.O. Box 37, Melbourne, FL 32901

The following should be inserted just above the statement labeled 430 in subroutine *EXCH*[1]:

```
Z(ITEMP) = Z(I)                EXC 1880
```

With this correction this algorithm has been used to fit first, second, and third degree curves to experimental data on the following machines: Harris SLASH 5, Honeywell 600 series, and Univac 1108.

[1] Subroutine *EXCH* is published in the "Collected Algorithms from ACM"; to insert line EXC 1880, see page 501-P8-R1

ALGORITHM 502
Dependence of Solution of Nonlinear Systems on a Parameter [C5]

MILAN KUBÍČEK

Prague Institute of Chemical Technology, Czechoslovakia

Key Words and Phrases: solution of nonlinear equation, differentiation with respect to a parameter, one-parameter imbedding, dependence on parameter
CR Categories: 5.15
Language: Fortran

DESCRIPTION

The Fortran algorithm *DERPAR* is a subroutine suitable for evaluation of the dependence of the solution $x_1, \ldots, x_n$ of a system of equations

$$\begin{aligned} f_1(x_1, \ldots, x_n, \alpha) &= 0 \\ &\vdots \\ f_n(x_1, \ldots, x_n, \alpha) &= 0 \end{aligned} \tag{1}$$

on the parameter α. The modified method of Daviděnko [1, 2, 3], which applies the implicit function theorem, is used in combination with the Newton method and Adams integration formulas. A similar procedure was used by Deist and Sefor [4], who made use of an introduced parameter.

To establish the dependence $x(\alpha)$, the following set of differential equations can be derived:

$$\frac{dx}{d\alpha} = -\Gamma^{-1}(x, \alpha)\frac{\partial f}{\partial \alpha}, \quad x(\alpha_0) = x_0. \tag{2}$$

Here

$$\Gamma = \left\{\frac{\partial f_i}{\partial x_j}\right\}, \quad \frac{\partial f}{\partial \alpha} = \left(\frac{\partial f_1}{\partial \alpha}, \ldots, \frac{\partial f_n}{\partial \alpha}\right)^{\mathrm{T}}, \quad f(x_0, \alpha_0) = 0.$$

If $\Gamma(x(\alpha), \alpha)$ is a regular matrix for $\langle \alpha_0, \alpha_1 \rangle$, then $x(\alpha)$ obtained by integration of (2) satisfies

$$f(x(\alpha), \alpha) = 0, \quad \alpha \in \langle \alpha_0, \alpha_1 \rangle. \tag{3}$$

However, at a branching point the matrix Γ is singular and the integration procedure mentioned above fails.

Therefore, the method must be adapted to be able to handle branching points, i.e. to evaluate the whole dependence $x(\alpha)$, which is a continuously smooth curve in the $(n + 1)$-th dimensional space $(x - \alpha)$. A parameterization with respect to

Received 27 December 1972 and 29 October 1973.

Author's address: Department of Chemical Engineering, Prague Institute of Chemical Technology, 166 28 Prague 6, Czechoslovakia.

the arc length of the solution locus seems to be a powerful technique which overcomes the difficulties [5].

Differentiation of eq. (1) with respect to t gives

$$\frac{df_i}{dt} = \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\frac{dx_j}{dt} + \frac{\partial f_i}{\partial \alpha}\frac{d\alpha}{dt} = 0, \quad i = 1, 2, \ldots, n. \tag{4}$$

An additional equation

$$\left(\frac{dx_1}{dt}\right)^2 + \cdots + \left(\frac{dx_n}{dt}\right)^2 + \left(\frac{d\alpha}{dt}\right)^2 = 1 \tag{5}$$

determines the parameter t as the arc length of the curve $x(\alpha)$ in the space $(x - \alpha)$. The initial conditions for eqs. (4) and (5) are in the form

$$t = 0 : x = x_0, \ \alpha = \alpha_0. \tag{6}$$

Equation (4) forms a system of n linear equations for $n+1$ unknowns dx_i/dt, $i = 1, \ldots, n$, $d\alpha/dt$, for each t. Suppose that the matrix

$$\Gamma_k = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}, \ldots, \frac{\partial f_1}{\partial x_{k-1}}, \frac{\partial f_1}{\partial x_{k+1}}, \ldots, \frac{\partial f_1}{\partial x_{n+1}} \\ \frac{\partial f_2}{\partial x_1}, \ldots \\ \vdots \\ \frac{\partial f_n}{\partial x_1}, \ldots, \frac{\partial f_n}{\partial x_{k-1}}, \frac{\partial f_n}{\partial x_{k+1}}, \ldots, \frac{\partial f_n}{\partial x_{n+1}} \end{pmatrix} \tag{7}$$

is regular (for certain values of t and k, $1 \leq k \leq n + 1$). For clarity here, we have denoted

$$x_{n+1} = \alpha. \tag{8}$$

Equation (4) can then be presolved with respect to the unknowns $dx_1/dt, \ldots, dx_{k-1}/dt, dx_{k+1}/dt, \ldots, dx_{n+1}/dt$, depending on dx_k/dt in the form

$$\frac{dx_i}{dt} = \beta_i \frac{dx_k}{dt}, \quad i = 1, 2, \ldots, k-1, k+1, \ldots, n+1. \tag{9}$$

In the program the unknowns with a subscript k (dx_k/dt and x_k) are called the independent variables. The value of the subscript k is determined in the Gaussian elimination procedure using controlled pivoting. The result depends on the values of $PREF(I)$.

On substituting (9) into (5) we obtain

$$\left(\frac{dx_k}{dt}\right)^2 = \left(1 + \sum_{i=1, i\neq k}^{n+1} \beta_i^2\right)^{-1}. \tag{10}$$

$\gamma = 1000$, $B = 22$, $\beta_1 = \beta_2 = 2$, $\theta_{c1} = \theta_{c2} = 0$; $x_0 = (0;0;0;0)$, $\alpha_0 = 0$,
$PREF(I) = (1,0.1,1,1,0.02)$, $HH = 0.02$, $EPS = 10^{-4}$, $MXADMS = 4$,
$NCORR = 4$, $NCRAD = 0$, $NDIR(I) = (1,1,1,1,1)$.
Calculated points on the curve $x_2(\alpha)$ are numbered. For the points 1–12, 61–70, 114–140, 155-. . ., α was independent ($k = 5$); otherwise x_2 ($k = 2$) plays the role of an independent variable. Altogether 332 evaluations of the subroutine *FCTN* for calculation of 162 points was necessary. However, by choosing $NCORR = 0$ and $NCRAD = 1$, the number of evaluations of *FCTN* was lowered to 163, and in fact the same curve was obtained.

The sign of dx_k/dt is given by the orientation of the parameter t along the curve. All derivatives dx_i/dt are then determined by (9).

The Adams-Bashforth explicit multistep method with an automatic change in the order of approximation is used for the integration of differential equations (9) and (10). It is possible to use some other technique, e.g. the Runge-Kutta method. However, it seems that our technique is more suitable when considering the time-

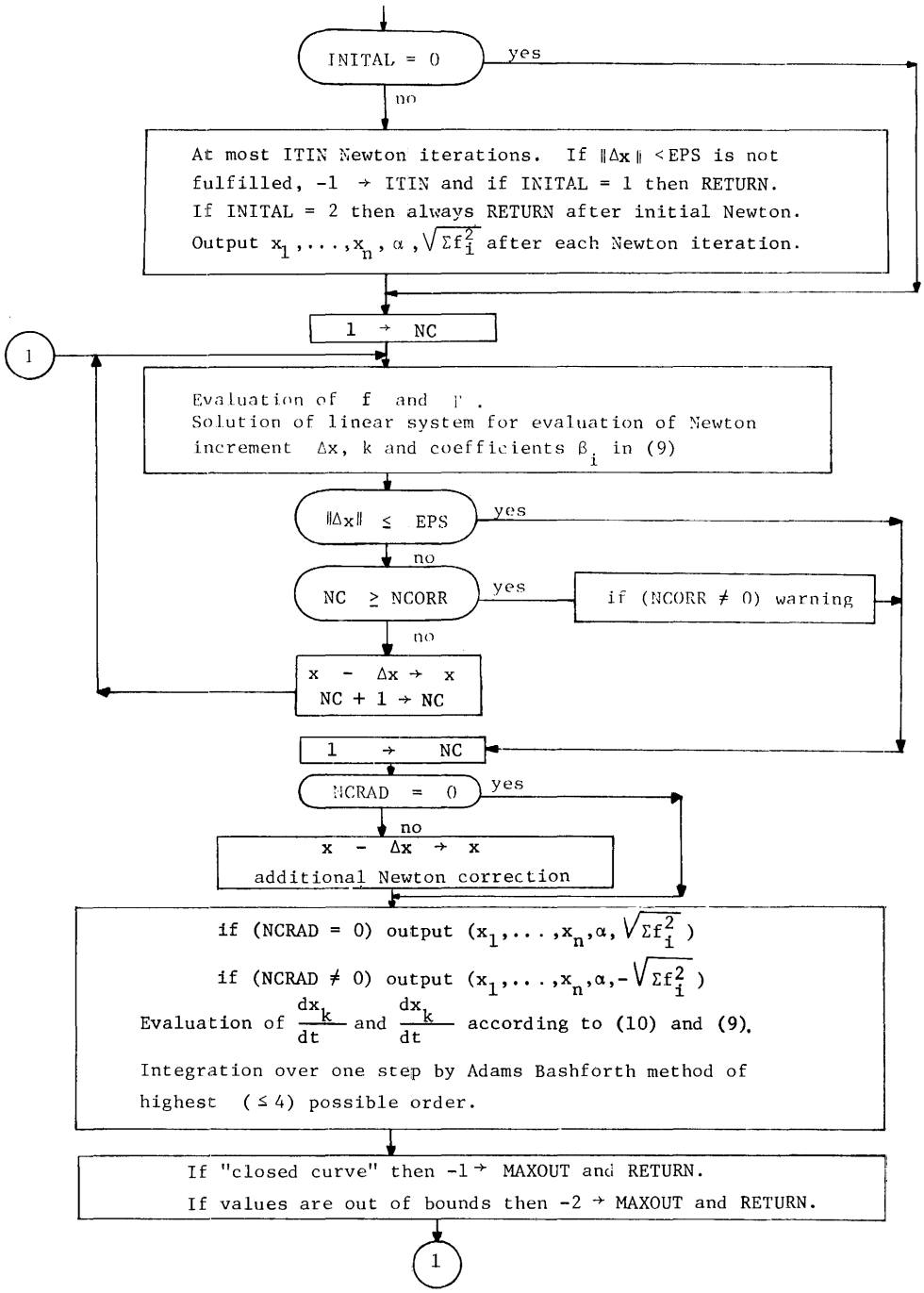


Fig. 1. Schematic flowchart of method

consuming evaluation of the Jacobian matrix and the matrix inversion procedure in each step.

In the course of integration the truncation error causes a deviation between the calculated solution $x(t)$ and the correct profiles $x(t)$. The Newton method for variables $x' = (x_1, x_2, \ldots, x_{k-1}, x_{k+1}, \ldots, x_{n+1})^T$ is therefore used to improve the calculated profiles:

$$x'_{\text{new}} - x'_{\text{old}} = -\Gamma_k^{-1} f.$$

The higher accuracy of numerical integration lowers the computer time expenditure only because of the low number of Newton iterations.

The algorithm is described by means of the schematic flow diagram given in

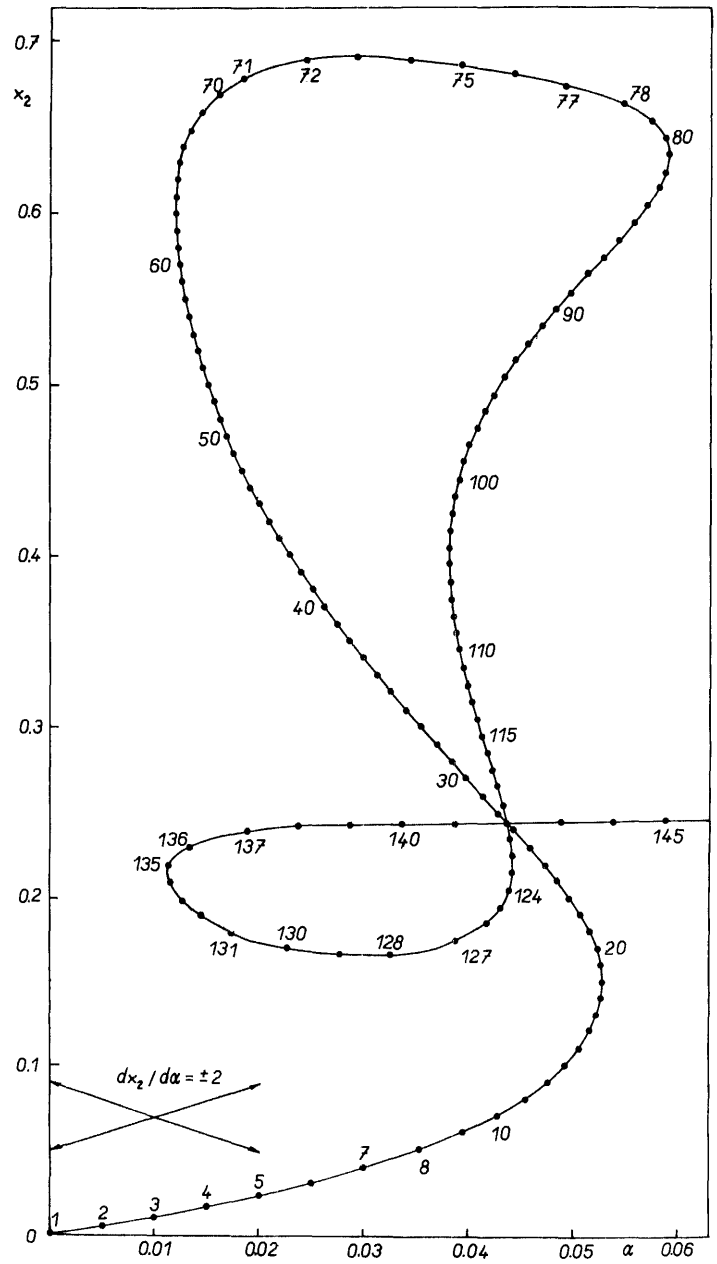


Fig. 2. The dependence $x_2(\alpha)$

Figure 1. The specification of the individual dummy (formal) parameters of the subroutine *DERPAR* is given in the subroutine heading.

The particular form of eq. (1) has to be programmed by the user as a subroutine *FCTN*. To illustrate the construction of the subroutine *FCTN*, an example is presented for a set of four nonlinear equations:

$$f_1 = \alpha(1 - x_3) \exp\,(10x_1/(1 + 10x_1/\gamma)) - x_3 = 0$$

$$f_2 = \alpha B(1 - x_3) \exp\,(10x_1/(1 + 10x_1/\gamma)) + \beta_1\theta_{c1} - 10(1 + \beta_1)x_1 = 0$$

$$f_3 = x_3 - x_4 + \alpha(1 - x_4) \exp\,(10x_2/(1 + 10x_2/\gamma)) = 0$$

$$f_4 = 10x_1 - 10(1 + \beta_2)x_2 + \alpha B(1 - x_4) \exp\,(10x_2/(1 + 10x_2/\gamma)) + \beta_2\theta_{c2} = 0.$$

Here γ, B, β_1, β_2, θ_{c1}, θ_{c2}, and α are physical parameters. The dependence $x_2(\alpha)$ is shown in Figure 2. Generally speaking, it is always convenient to transform the unknowns $x_1, x_2, \ldots, x_n, \alpha$ so that their values are of similar magnitude.

Applicability. This program can be used in the computation of the dependence of a solution on a parameter for sets of nonlinear equations, for nonlinear boundary value problems for ordinary differential equations [6, 7], for difference equations [8], etc.

ACKNOWLEDGMENTS

The author would like to thank the referee for his very valuable comments.

REFERENCES

[1] DAVIDĚNDO, D.F. On a new method of numerical solution of systems of nonlinear equations. *Dokl. Akad. Nauk SSSR 88*(1953), 601–602.
[2] FICKEN, F.A. The continuation method for functional equations. *Comm. Pure Appl. Math. 4*(1951), 435–456.
[3] HASELGROVE, C.B. The solution of nonlinear equations with two-point boundary conditions. *Computer J. 4*(1961), 255–259.
[4] DEIST, F.H., and SEFOR, L. Solutions of systems of nonlinear equations by parameter variation. *Computer J. 10*(1967), 78–82.
[5] KLOPFENSTEIN, R.W. Zeros of nonlinear functions. *J. ACM 8*(1961), 366–373.
[6] KUBÍČEK, M., and HLAVÁČEK, V. Solution of nonlinear boundary value problems: Va. A novel method, general parameter mapping (GPM); Vb. Predictor-corrector GPM method. *Chem. Eng. Sci. 27*(1972), 743–750 and 2095–2098.
[7] KUBÍČEK, M., and HLAVÁČEK, V. General parameter mapping technique—a procedure for solution of non-linear boundary value problems depending on an actual parameter. *J. Inst. Math. Appl. 12*(1973), 287–293.
[8] KUBÍČEK, M., HLAVÁČEK, V., and JELÍNEK, J. Solution of counter-current separation processes: IV. Application of the GPM method to solution of distillation problems. *Chem. Eng. Sci. 29*(1974), 435–441.

ALGORITHM

```
      SUBROUTINE DERPAR(N, X, XLOW, XUPP, EPS, W, INITAL, ITIN, HH,     DER   10
     * HMAX, PREF, NDIR, E, MXADMS, NCORR, NCRAD, NOUT, OUT,            DER   20
     * MAXOUT, NPRNT)                                                   DER   30
      DIMENSION X(11), XLOW(11), XUPP(11), W(11), HMAX(11),             DER   40
     * PREF(11), NDIR(11), OUT(100,12), F(11), G(10,11), BETA(11),      DER   50
     * MARK(11), DXDT(11)                                               DER   60
C OBTAINING OF DEPENDENCE OF SOLUTION X(ALFA) OF EQUATION               DER   70
C F ( X , ALFA ) = 0                                                    DER   80
C ON PARAMETER ALFA BY MODIFIED METHOD OF DIFFERENTIATION               DER   90
C WITH RESPECT TO PARAMETER                                             DER  100
C N - NUMBER OF UNKNOWNS X(I)                                           DER  110
C X(1),...,X(N) - INITIAL VALUES OF X(I), AFTER RETURN FINAL VALUES     DER  120
C                 OF X(I)                                               DER  130
C X(N+1) - INITIAL VALUE OF PARAMETER ALFA, AFTER RETURN FINAL VALUE    DER  140
C          OF ALFA                                                      DER  150
C XLOW(1),...,XLOW(N) - LOWER BOUNDS FOR X(I)                           DER  160
C XUPP(1),...,XUPP(N) - UPPER BOUNDS FOR X(I)                           DER  170
C XLOW(N+1),XUPP(N+1) - LOWER AND UPPER BOUNDS FOR ALFA                 DER  180
C    IF XLOW OR XUPP IS EXCEEDED, THEN END OF DERPAR                    DER  190
C    AND MAXOUT=-2 AFTER RETURN                                         DER  200
C EPS - ACCURACY DESIRED IN NEWTON ITERATION FOR                        DER  210
C    SUM OF (W(I)*ABS(XNEW(I)-XOLD(I))), I=1,...,N+1                    DER  220
C W(1),...,W(N+1) - WEIGHTS USED IN TERMINATION CRITERION OF NEWTON     DER  230
C                   PROCESS                                             DER  240
C INITAL - IF (INITAL.NE.0) THEN SEVERAL STEPS IN NEWTON ITERATION      DER  250
C          ARE MADE BEFORE COMPUTATION IN ORDER TO INCREASE             DER  260
C          ACCURACY OF INITIAL POINT-                                   DER  270
C     IF (INITAL.EQ.1.AND.EPS-ACCURACY IS NOT FULFILLED IN ITIN         DER  280
C     ITERATIONS) THEN RETURN.   IF (INITAL.EQ.2) THEN ALWAYS RETURN    DER  290
C     AFTER INITIAL NEWTON ITERATION, RESULTS ARE IN X.                 DER  300
C     IF (INITAL.EQ.3) THEN CONTINUE IN DERPAR AFTER INITIAL NEWTON.    DER  310
C     IF (INITAL.EQ.0) THEN NO INITIAL NEWTON ITERATION IS USED.        DER  320
C ITIN - MAXIMAL NUMBER OF INITIAL NEWTON ITERATIONS. IF                DER  330
C        EPS-ACCURACY IS NOT FULFILLED IN ITIN ITERATIONS THEN          DER  340
C        ITIN=-1 AFTER RETURN.                                          DER  350
C HH - INTEGRATION STEP ALONG ARC LENGTH OF SOLUTION LOCUS              DER  360
C HMAX(1),...,HMAX(N+1) - UPPER BOUNDS FOR INCREMENTS OF X(I) IN        DER  370
C                         ONE INTEGRATION STEP (APPROXIMATION ONLY)     DER  380
C PREF(1),...,PREF(N+1) - PREFERENCE NUMBERS (EXPLANATION SEE IN        DER  390
C                         SUBR.GAUSE)                                   DER  400
C NDIR(1),...,NDIR(N+1) - INITIAL CHANGE OF X(I) IS POSITIVE ALONG      DER  410
C     SOLUTION LOCUS (CURVE) IF NDIR(I)=1 AND NEGATIVE IF               DER  420
C     NDIR(I)=-1.                                                       DER  430
C E - CRITERION FOR TEST ON CLOSED CURVE, IF                            DER  440
C    (SUM OF (W(I)*ABS(X(I)-XINITIAL(I))),I=1,...,N+1).LE.E)            DER  450
C    THEN CLOSED CURVE MAY BE EXPECTED.                                 DER  460
C MXADMS - MAXIMAL ORDER OF ADAMS-BASHFORTH FORMULA,                    DER  470
C          1.LE.MXADMS.LE.4.                                            DER  480
```

```
C NCORR - MAXIMAL NUMBER OF NEWTON CORRECTIONS AFTER PREDICTION          DER  490
C    BY ADAMS-BASHFORTH METHOD.                                          DER  500
C NCRAD - IF (NCRAD.NE.0) THEN ADDITIONAL NEWTON CORRECTION              DER  510
C    WITHOUT NEW COMPUTATION OF JACOBIAN MATRIX IS USED.                 DER  520
C NOUT - AFTER RETURN NUMBER OF CALCULATED POINTS ON THE CURVE           DER  530
C        X(ALFA), NOUT.LE.MAXOUT.                                        DER  540
C OUT(J,1),OUT(J,2),...,OUT(J,N+1) - J-TH POINT X(1),...,X(N),ALFA       DER  550
C    ON CURVE X(ALFA)                                                    DER  560
C OUT(J,N+2) - VALUE OF SQRT(SUM OF SQUARES OF F).                       DER  570
C    IF (OUT(J,N+2).LT.0.0) THEN  ABS(OUT(J,N+2)) CORRESPONDS TO X       DER  580
C    AND ALFA NOT EXACTLY, BECAUSE ADDITIONAL NEWTON CORRECTION WAS      DER  590
C    USED (NCRAD.NE.0).                                                  DER  600
C    VALUES F(I) ARE NOT COMPUTED FOR X AND ALFA PRINTED AND/OR          DER  610
C    STORED AND THEREFORE LAST TIME COMPUTED VALUE OF SQRT(SUM OF        DER  620
C    SQUARES OF F) IS ON OUR DISPOSAL ONLY.                              DER  630
C MAXOUT - MAXIMAL NUMBER OF CALCULATED POINTS ON CURVE X(ALFA).         DER  640
C    IF MAXOUT AFTER RETURN EQUALS TO-                                   DER  650
C          -1 - THEN CLOSED CURVE X(ALFA) MAY BE EXPECTED                DER  660
C          -2 - THEN BOUND XLOW OR XUPP WAS EXCEEDED                     DER  670
C          -3 - THEN SINGULAR JACOBIAN MATRIX OCCURRED, ITS RANK IS      DER  680
C               .LT. N.                                                  DER  690
C NPRNT - IF (NPRNT.EQ.3) THEN RESULTING POINTS ON CURVE X(ALFA)         DER  700
C    ARE IN ARRAY OUT( , ) AFTER RETURN. IF (NPRNT.EQ.1) THEN THESE      DER  710
C    POINTS ARE PRINTED ONLY.  IF (NPRNT.EQ.2) THEN THESE POINTS ARE     DER  720
C    BOTH PRINTED AND IN ARRAY OUT.                                      DER  730
C SUBROUTINE FCTN MUST BE PROGRAMMED IN FOLLOWING FORM -                 DER  740
C   SUBROUTINE FCTN(N,X,F,G)                                             DER  750
C   DIMENSION X(11),F(10),G(10,11)                                       DER  760
C   F(I)= FI (X(1),X(2),...,X(N),ALFA) FOR I=1,...,N                     DER  770
C   G(I,J)= D FI (X(1),...,X(N),ALFA)/ D X(J) FOR  I,J=1,...,N           DER  780
C   G(I,N+1)= D FI (X(1),...,X(N),ALFA)/ D ALFA  FOR I=1,...,N           DER  790
C   RETURN                                                               DER  800
C   END                                                                  DER  810
      DATA INDIC, INDSP /1H*,1H /                                        DER  820
C LW IS PRINTER DEVICE NUMBER                                            DER  830
      N1 = N + 1                                                         DER  840
      LW = 3                                                             DER  850
      IF (INITAL) 10, 60, 10                                             DER  860
C INITIAL NEWTON ITERATIONS                                              DER  870
   10 DO 40 L=1,ITIN                                                     DER  880
        CALL FCTN(N, X, F, G)                                            DER  890
        SQUAR = 0.0                                                      DER  900
        DO 20 I=1,N                                                      DER  910
          SQUAR = SQUAR + F(I)**2                                        DER  920
   20   CONTINUE                                                         DER  930
        LL = L - 1                                                       DER  940
        SQUAR = SQRT(SQUAR)                                              DER  950
        IF (NPRNT.NE.3) WRITE (LW,99999) LL, (X(I),I=1,N1), SQUAR        DER  960
        CALL GAUSE(N, G, F, M, 10, 11, PREF, BETA, K)                    DER  970
        IF (M.EQ.0) GO TO 310                                            DER  980
        P = 0.0                                                          DER  990
        DO 30 J=1,N1                                                     DER 1000
          X(J) = X(J) - F(J)                                             DER 1010
          P = P + ABS(F(J))*W(J)                                         DER 1020
   30   CONTINUE                                                         DER 1030
        IF (P.LE.EPS) GO TO 50                                           DER 1040
   40 CONTINUE                                                           DER 1050
      WRITE (LW,99998) ITIN                                              DER 1060
      ITIN = -1                                                          DER 1070
      IF (INITAL.EQ.1) RETURN                                            DER 1080
   50 IF (NPRNT.NE.3) WRITE (LW,99997) (X(I),I=1,N1)                     DER 1090
      IF (INITAL.EQ.2) RETURN                                            DER 1100
C AFTER INITIAL NEWTON ITERATIONS                                        DER 1110
   60 IF (NPRNT.NE.3) WRITE (LW,99996)                                   DER 1120
      KOUT = 0                                                           DER 1130
      NOUT = 0                                                           DER 1140
      MADMS = 0                                                          DER 1150
      NC = 1                                                             DER 1160
      K1 = 0                                                             DER 1170
   70 CALL FCTN(N, X, F, G)                                              DER 1180
      SQUAR = 0.0                                                        DER 1190
      DO 80 I=1,N                                                        DER 1200
        SQUAR = SQUAR + F(I)**2                                          DER 1210
   80 CONTINUE                                                           DER 1220
      CALL GAUSE(N, G, F, M, 10, 11, PREF, BETA, K)                      DER 1230
      IF (M.EQ.0) GO TO 310                                              DER 1240
      IF (K1.EQ.K) GO TO 90                                              DER 1250
C CHANGE OF INDEPENDENT VARIABLE (ITS INDEX = K NOW)                     DER 1260
      MADMS = 0                                                          DER 1270
      K1 = K                                                             DER 1280
   90 SQUAR = SQRT(SQUAR)                                                DER 1290
      IF (NCRAD.EQ.1) SQUAR = -SQUAR                                     DER 1300
      P = 0.0                                                            DER 1310
      DO 100 I=1,N1                                                      DER 1320
        P = P + W(I)*ABS(F(I))                                           DER 1330
  100 CONTINUE                                                           DER 1340
      IF (P.LE.EPS) GO TO 130                                            DER 1350
      IF (NC.GE.NCORR) GO TO 120                                         DER 1360
      DO 110 I=1,N1                                                      DER 1370
        X(I) = X(I) - F(I)                                               DER 1380
  110 CONTINUE                                                           DER 1390
C ONE ITERATION IN NEWTON METHOD                                         DER 1400
```

```
      NC = NC + 1                                                        DER 1410
      GO TO 70                                                           DER 1420
  120 IF (NCORR.EQ.0) GO TO 130                                          DER 1430
      WRITE (LW,99995) NCORR, P                                          DER 1440
  130 NC = 1                                                             DER 1450
      IF (NCRAD.EQ.0) GO TO 150                                          DER 1460
C ADDITIONAL NEWTON CORRECTION                                           DER 1470
      DO 140 I=1,N1                                                      DER 1480
        X(I) = X(I) - F(I)                                               DER 1490
  140 CONTINUE                                                           DER 1500
  150 NOUT = NOUT + 1                                                    DER 1510
      DO 160 I=1,N1                                                      DER 1520
        MARK(I) = INDSP                                                  DER 1530
  160 CONTINUE                                                           DER 1540
      MARK(K) = INDIC                                                    DER 1550
      IF (NPRNT.EQ.3) GO TO 170                                          DER 1560
      WRITE (LW,99994)                                                   DER 1570
      WRITE (LW,99993) (X(I),MARK(I),I=1,N1), SQUAR                      DER 1580
  170 IF (NPRNT.EQ.1) GO TO 200                                          DER 1590
      IF (NOUT.LE.100) GO TO 180                                         DER 1600
      WRITE (LW,99992)                                                   DER 1610
      RETURN                                                             DER 1620
  180 DO 190 I=1,N1                                                      DER 1630
        OUT(NOUT,I) = X(I)                                               DER 1640
  190 CONTINUE                                                           DER 1650
      OUT(NOUT,N+2) = SQUAR                                              DER 1660
      GO TO 210                                                          DER 1670
  200 IF (NOUT.EQ.1) GO TO 180                                           DER 1680
  210 IF (NOUT.GE.MAXOUT) RETURN                                         DER 1690
      DO 220 I=1,N1                                                      DER 1700
        IF (X(I).LT.XLOW(I) .OR. X(I).GT.XUPP(I)) GO TO 300              DER 1710
  220 CONTINUE                                                           DER 1720
      IF (NOUT.LE.3) GO TO 240                                           DER 1730
      P = 0.0                                                            DER 1740
      DO 230 I=1,N1                                                      DER 1750
        P = P + W(I)*ABS(X(I)-OUT(1,I))                                  DER 1760
  230 CONTINUE                                                           DER 1770
      IF (P.LE.E) GO TO 290                                              DER 1780
C CLOSED CURVE MAY BE EXPECTED                                           DER 1790
  240 DXK2 = 1.0                                                         DER 1800
      DO 250 I=1,N1                                                      DER 1810
        DXK2 = DXK2 + BETA(I)**2                                         DER 1820
  250 CONTINUE                                                           DER 1830
      DXDT(K) = 1.0/SQRT(DXK2)*FLOAT(NDIR(K))                            DER 1840
C DERIVATIVE OF INDEPENDENT VARIABLE X(K) WITH RESPECT TO ARC LENGTH     DER 1850
C OF SOLUTION LOCUS IS COMPUTED HERE                                     DER 1860
      H = HH                                                             DER 1870
      DO 270 I=1,N1                                                      DER 1880
        NDIR(I) = 1                                                      DER 1890
        IF (I.EQ.K) GO TO 260                                            DER 1900
        DXDT(I) = BETA(I)*DXDT(K)                                        DER 1910
  260   IF (DXDT(I).LT.0.0) NDIR(I) = -1                                 DER 1920
        IF (H*ABS(DXDT(I)).LE.HMAX(I)) GO TO 270                         DER 1930
        MADMS = 0                                                        DER 1940
        H = HMAX(I)/ABS(DXDT(I))                                         DER 1950
  270 CONTINUE                                                           DER 1960
      IF (NOUT.LE.KOUT+3) GO TO 280                                      DER 1970
      IF (H*ABS(DXDT(K)).LE.0.8*ABS(X(K)-OUT(1,K))) GO TO 280            DER 1980
      IF ((OUT(1,K)-X(K))*FLOAT(NDIR(K)).LE.0.0) GO TO 280               DER 1990
      MADMS = 0                                                          DER 2000
      IF (H*ABS(DXDT(K)).LE.ABS(X(K)-OUT(1,K))) GO TO 280                DER 2010
      H = ABS(X(K)-OUT(1,K))/ABS(DXDT(K))                                DER 2020
      KOUT = NOUT                                                        DER 2030
  280 CALL ADAMS(N, DXDT, MADMS, H, X, MXADMS)                           DER 2040
      GO TO 70                                                           DER 2050
  290 WRITE (LW,99991)                                                   DER 2060
      MAXOUT = -1                                                        DER 2070
      RETURN                                                             DER 2080
  300 MAXOUT = -2                                                        DER 2090
      RETURN                                                             DER 2100
  310 WRITE (LW,99990) (X(I),I=1,N1)                                     DER 2110
      MAXOUT = -3                                                        DER 2120
      RETURN                                                             DER 2130
99999 FORMAT (3X, 7H DERPAR, I3, 25H.INITIAL NEWTON ITERATION/22X,       DER 2140
     * 11HX,ALFA,SQF=, 5F15.7/(33X, 5F15.7))                             DER 2150
99998 FORMAT (/16H DERPAR OVERFLOW, I5, 2X, 18HINITIAL ITERATIONS/)       DER 2160
99997 FORMAT (3X, 39H DERPAR AFTER INITIAL NEWTON ITERATIONS/22X,         DER 2170
     * 7HX,ALFA=, 4X, 5F15.7/(33X, 5F15.7))                              DER 2180
99996 FORMAT (/6X, 45H DERPAR RESULTS (VARIABLE CHOSEN AS INDEPENDE,      DER 2190
     * 18HNT IS MARKED BY *)/)                                           DER 2200
99995 FORMAT (/37H DERPAR NUMBER OF NEWTON CORRECTIONS=, I5,              DER 2210
     * 31H  IS NOT SUFFICIENT,ERROR OF X=, F15.7/)                       DER 2220
99994 FORMAT (6X, 27H DERPAR RESULTS X,ALFA,SQF=)                        DER 2230
99993 FORMAT (33X, 5(F15.7, A1))                                         DER 2240
99992 FORMAT (/28H DERPAR OUTPUT ARRAY IS FULL//)                        DER 2250
99991 FORMAT (/36H DERPAR CLOSED CURVE MAY BE EXPECTED/)                 DER 2260
99990 FORMAT (/48H DERPAR SINGULAR JACOBIAN MATRIX FOR X AND ALFA=,      DER 2270
     * 5F12.6/(48X, 5F12.6))                                            DER 2280
      END                                                                DER 2290

      SUBROUTINE GAUSE(N, A, B, M, NN, MM, PREF, BETA, K)                GAU   10
      DIMENSION A(NN,MM), B(MM), PREF(MM), BETA(MM), Y(11), X(11),       GAU   20
```

```
     * IRR(11), IRK(11)                                                GAU  30
C SOLUTION OF N LINEAR EQUATIONS FOR N+1 UNKNOWNS                      GAU  40
C BASED ON GAUSSIAN ELIMINATION WITH PIVOTING.                         GAU  50
C N - NUMBER OF EQUATIONS                                              GAU  60
C A - N X (N+1) MATRIX OF SYSTEM                                       GAU  70
C B - RIGHT-HAND SIDES                                                 GAU  80
C M - IF (M.EQ.0) AFTER RETURN THEN RANK(A).LT.N                       GAU  90
C PREF(I) - PREFERENCE NUMBER FOR X(I) TO BE INDEPENDENT VARIABLE,     GAU 100
C 0.0.LE.PREF(I).LE.1.0, THE LOWER IS PREF(I) THE HIGHER IS            GAU 110
C PREFERENCE OF X(I).                                                  GAU 120
C BETA(I) - COEFFICIENTS IN EXPLICIT DEPENDENCES OBTAINED IN FORM -    GAU 130
C          X(I)=B(I)+BETA(I)*X(K), I.NE.K.                             GAU 140
C K - RESULTING INDEX OF INDEPENDENT VARIABLE                          GAU 150
      N1 = N + 1                                                       GAU 160
      ID = 1                                                           GAU 170
      M = 1                                                            GAU 180
      DO 10 I=1,N1                                                     GAU 190
        IRK(I) = 0                                                     GAU 200
        IRR(I) = 0                                                     GAU 210
   10 CONTINUE                                                         GAU 220
   20 IR = 1                                                           GAU 230
      IS = 1                                                           GAU 240
      AMAX = 0.0                                                       GAU 250
      DO 60 I=1,N                                                      GAU 260
        IF (IRR(I)) 60, 30, 60                                         GAU 270
   30   DO 50 J=1,N1                                                   GAU 280
          P = PREF(J)*ABS(A(I,J))                                      GAU 290
          IF (P-AMAX) 50, 50, 40                                       GAU 300
   40     IR = I                                                       GAU 310
          IS = J                                                       GAU 320
          AMAX = P                                                     GAU 330
   50   CONTINUE                                                       GAU 340
   60 CONTINUE                                                         GAU 350
      IF (AMAX.NE.0.0) GO TO 70                                        GAU 360
C THIS CONDITION FOR SINGULARITY OF MATRIX MUST BE SPECIFIED           GAU 370
C MORE EXACTLY WITH RESPECT TO COMPUTER ACTUALLY USED                  GAU 380
      M = 0                                                            GAU 390
      GO TO 150                                                        GAU 400
   70 IRR(IR) = IS                                                     GAU 410
      DO 90 I=1,N                                                      GAU 420
        IF (I.EQ.IR .OR. A(I,IS).EQ.0.0) GO TO 90                      GAU 430
        P = A(I,IS)/A(IR,IS)                                           GAU 440
        DO 80 J=1,N1                                                   GAU 450
          A(I,J) = A(I,J) - P*A(IR,J)                                  GAU 460
   80   CONTINUE                                                       GAU 470
        A(I,IS) = 0.0                                                  GAU 480
        B(I) = B(I) - P*B(IR)                                          GAU 490
   90 CONTINUE                                                         GAU 500
      ID = ID + 1                                                      GAU 510
      IF (ID.LE.N) GO TO 20                                            GAU 520
      DO 100 I=1,N                                                     GAU 530
        IR = IRR(I)                                                    GAU 540
        X(IR) = B(I)/A(I,IR)                                           GAU 550
        IRK(IR) = 1                                                    GAU 560
  100 CONTINUE                                                         GAU 570
      DO 110 K=1,N1                                                    GAU 580
        IF (IRK(K).EQ.0) GO TO 120                                     GAU 590
  110 CONTINUE                                                         GAU 600
  120 DO 130 I=1,N                                                     GAU 610
        IR = IRR(I)                                                    GAU 620
        Y(IR) = -A(I,K)/A(I,IR)                                        GAU 630
  130 CONTINUE                                                         GAU 640
      DO 140 I=1,N1                                                    GAU 650
        B(I) = X(I)                                                    GAU 660
        BETA(I) = Y(I)                                                 GAU 670
  140 CONTINUE                                                         GAU 680
      B(K) = 0.0                                                       GAU 690
      BETA(K) = 0.0                                                    GAU 700
  150 RETURN                                                           GAU 710
      END                                                              GAU 720


      SUBROUTINE ADAMS(N, D, MADMS, H, X, MXADMS)                      ADA  10
      DIMENSION DER(4,11), X(11), D(11)                                ADA  20
C ADAMS-BASHFORTH METHODS                                              ADA  30
      N1 = N + 1                                                       ADA  40
      DO 20 I=1,3                                                      ADA  50
        DO 10 J=1,N1                                                   ADA  60
          DER(I+1,J) = DER(I,J)                                        ADA  70
   10   CONTINUE                                                       ADA  80
   20 CONTINUE                                                         ADA  90
      MADMS = MADMS + 1                                                ADA 100
      IF (MADMS.GT.MXADMS) MADMS = MXADMS                              ADA 110
      IF (MADMS.GT.4) MADMS = 4                                        ADA 120
      DO 70 I=1,N1                                                     ADA 130
        DER(1,I) = D(I)                                                ADA 140
        GO TO (30, 40, 50, 60), MADMS                                  ADA 150
   30   X(I) = X(I) + H*DER(1,I)                                       ADA 160
        GO TO 70                                                       ADA 170
   40   X(I) = X(I) + 0.5*H*(3.0*DER(1,I)-DER(2,I))                    ADA 180
        GO TO 70                                                       ADA 190
```

```
   50   X(I) = X(I) + H*(23.0*DER(1,I)-16.0*DER(2,I)+5.0*DER(3,I))/      ADA  200
     *    12.0                                                          ADA  210
          GO TO 70                                                      ADA  220
60    X(I) = X(I) + H*(55.0*DER(1,I)-59.0*DER(2,I)+37.0*DER(3,I)        ADA  230
  *    -9.0*DER(4,I))/24.0                                             ADA  240
70 CONTINUE                                                           ADA  250
   RETURN                                                             ADA  260
   END                                                                ADA  270
```

ALGORITHM 503
An Automatic Program for Fredholm Integral Equations of the Second Kind [D5]

KENDALL ATKINSON

University of Iowa

Key Words and Phrases: numerical analysis, linear integral equations, automatic algorithm, Nyström method
CR Categories: 5.18
Language: Fortran

DESCRIPTION

The two algorithms given here, *IESIMP* and *IEGAUS*, are a complement of [1] where the theoretical development is described.

REFERENCES

1. ATKINSON, K. An automatic program for Fredholm integral equations of the second kind. *ACM Trans. Math. Software 2*, 2 (June 1976), 154–171.

ALGORITHMS

```
      SUBROUTINE IESIMP(KERNEL, RHFCN, A, B, NZ, EPS, IFLAG, X,          IES   1Ø
     * NUPPER, MUPPER, W, IFLGR2, IER)                                   IES   2Ø
C THE INTEGRAL EQUATION BEING SOLVED IS                                  IES   3Ø
C                    B                                                   IES   4Ø
C        X(S) -  INT  KERNEL(S,T)*X(T)*DT = RHFCN(S)                     IES   5Ø
C                   A                                                    IES   6Ø
C THE METHOD BEING USED IS BASED ON THE NYSTROM METHOD WITH              IES   7Ø
C SIMPSONS RULE, WITH AN ITERATIVE TECHNIQUE OF SOLUTION FOR THE         IES   8Ø
C RESULTING LINEAR SYSTEM.                                               IES   9Ø
C KERNEL     THESE ARE DOUBLE PRECISION FUNCTIONS OF TWO AND ONE         IES  1ØØ
C RHFCN      VARIABLES, RESPECTIVELY. THEY MUST BE DECLARED IN AN        IES  11Ø
C            EXTERNAL STATEMENT IN THE PROGRAM CALLING IESIMP.           IES  12Ø
C NZ         THE INITIAL VALUE OF N IN THE PROGRAM. N IS THE ORDER       IES  13Ø
C            OF AN INVERSE MATRIX WHICH IS BEING USED TO                 IES  14Ø
C            ITERATIVELY SOLVE A LARGER SYSTEM OF ORDER M, WHICH         IES  15Ø
C            APPROXIMATES THE ABOVE INTEGRAL EQUATION. THE CHOICE        IES  16Ø
C            OF NZ MUST ALWAYS BE ODD AND GREATER THAN TWO. FOR A        IES  17Ø
C            DEFAULT CHOICE, SET NZ=3.                                   IES  18Ø
C            ON COMPLETION OF THE PROGRAM, NZ IS SET EQUAL TO THE        IES  19Ø
C            FINAL NUMBER OF NODE POINTS USED IN OBTAINING THE           IES  2ØØ
C            SOLUTION. THE ARRAYS W AND X WILL CONTAIN THE NZ            IES  21Ø
C            FINAL NODE POINTS AND CORRESPONDING SOLUTION                IES  22Ø
C            VALUES.                                                     IES  23Ø
C EPS        THE DESIRED ERROR, INTERPRETED ACCORDING TO IFLAG.          IES  24Ø
```

Received 12 June 1974 and 22 July 1975.

Author's address: Department of Mathematics, University of Iowa, Iowa City, IA 52242.

```
C             THIS VARIABLE IS CHANGED ON COMPLETION OF THE              IES  25Ø
C             PROGRAM. SEE THE DISCUSSION OF IFLAG AND IER FOR MORE      IES  26Ø
C             INFORMATION.                                               IES  27Ø
C IFLAG =Ø    EPS IS TO BE INTERPRETED AS AN ABSOLUTE ERROR.             IES  28Ø
C       =1    EPS IS TO BE INTERPRETED AS A RELATIVE ERROR.              IES  29Ø
C X           THE ANSWER TO THE INTEGRAL EQUATION, EVALUATED AT THE      IES  3ØØ
C             FINAL SET OF NODE POINTS, WHICH ARE STORED IN W. THE       IES  31Ø
C             DIMENSION OF X SHOULD BE AT LEAST MUPPER.                  IES  32Ø
C NUPPER      AN UPPER LIMIT ON THE VARIABLE N IN THIS PROGRAM. N        IES  33Ø
C             IS THE ORDER OF APPROXIMATE INVERSE WHICH IS BEING         IES  34Ø
C             USED TO ITERATIVELY SOLVE A LARGER SYSTEM OF ORDER M.      IES  35Ø
C MUPPER      AN UPPER LIMIT ON THE VARIABLE M IN THE PROGRAM.           IES  36Ø
C             M IS THE NUMBER OF NODE POINTS BEING USED IN THE           IES  37Ø
C             SIMPSON RULE APPROXIMATION OF THE INTEGRAL OPERATOR.       IES  38Ø
C             N AND M ARE ALWAYS ODD INTEGERS, GREATER THAN TWO.         IES  39Ø
C W           THIS IS TEMPORARY WORKING STORAGE. IT SHOULD HAVE          IES  4ØØ
C             DIMENSION 4*NU*NU+1Ø*NU+9*MU, WITH NU=NUPPER,              IES  41Ø
C             MU=MUPPER. ON EXIT, W WILL CONTAIN THE FINAL CHOICE        IES  42Ø
C             OF NODE POINTS,CORRESPONDING TO THE APPROXIMATE            IES  43Ø
C             SOLUTION VALUES STORED IN X.                               IES  44Ø
C IFLGR2 =Ø   THE NORMAL SETTING.                                        IES  45Ø
C        =1   IF THE INTEGRAND K(S,T)*X(T), REGARDED AS A FUNCTION       IES  46Ø
C             OF T, IS KNOWN TO BE PERIODIC ALONG WITH A NUMBER OF       IES  47Ø
C             ITS DERIVATIVES ON THE INTERVAL (A,B), FOR ALL S IN        IES  48Ø
C             THE INTERVAL, THEN THE ERROR ESTIMATES WILL BE MORE        IES  49Ø
C             ACCURATE IF IFLGR2=1.  THIS WILL GENERALLY RESULT IN       IES  5ØØ
C             MORE RAPID CONVERGENCE OF THE PROGRAM.                     IES  51Ø
C IER =Ø      THIS ERROR COMPLETION CODE MEANS THAT THE ERROR TEST       IES  52Ø
C             WAS SATISFIED. THE PREDICTED ERROR IS STORED IN EPS.       IES  53Ø
C     =1      THE ERROR TEST WAS NOT SATISFIED. THE PREDICTED ERROR      IES  54Ø
C             IS IN EPS.                                                 IES  55Ø
C     =2      THE ERROR TEST WAS NOT SATISFIED. THE VARIABLE EPS         IES  56Ø
C             HAS BEEN SET TO ZERO.                                      IES  57Ø
C             REGARDLESS OF THE VALUE OF IER, THE MOST ACCURATE          IES  58Ø
C             SOLUTION OBTAINED IS STORED IN X.                          IES  59Ø
C IESIMP IS PRESENTLY LIMITED TO NUPPER .LE. 1ØØ. THIS IS ENTIRELY       IES  6ØØ
C DUE TO A RESTRICTION IN THE PROGRAM LINSYS. TO REMOVE THIS             IES  61Ø
C RESTRICTION, CHANGE THE DIMENSION STATEMENT IN THE                     IES  62Ø
C COMMON/LINEAR/ STATEMENT IN LINSYS.                                    IES  63Ø
      DOUBLE PRECISION KERNEL, RHFCN, A, B, EPS, X, W, CUTOFF,           IES  64Ø
     * RATIO, ROOTRT                                                     IES  65Ø
      DIMENSION X(1), W(1)                                               IES  66Ø
      EXTERNAL KERNEL, RHFCN                                             IES  67Ø
C ************************************************************************IES  68Ø
C                                                                       *  IES  69Ø
      COMMON /INFO/ R1, R2, NFINAL                                       IES  7ØØ
C                                                                       *  IES  71Ø
C    THE VARIABLES IN COMMON/INFO/ GIVE ADDITIONAL INFORMATION ABOUT    *  IES  72Ø
C    THE FUNCTIONING OF IESIMP. R1 GIVES THE FINAL ITERATIVE RATE       *  IES  73Ø
C    OF CONVERGENCE FOR SOLVING THE LINEAR SYSTEMS OF ORDER M. R2       *  IES  74Ø
C    GIVES THE RATE OF CONVERGENCE OF THE NYSTROM METHOD. FOR A         *  IES  75Ø
C    SMOOTH KERNEL FUNCTION AND SMOOTH SOLUTION FUNCTION X(S), THE      *  IES  76Ø
C    VALUE OF R2 SHOULD BE Ø.Ø625 FOR SIMPSONS RULE.  NFINAL GIVES      *  IES  77Ø
C    THE FINAL VALUE OF N USED IN ITERATIVELY SOLVING THE LARGER        *  IES  78Ø
C    SYSTEMS OF ORDER M. IF THE VALUE OF NFINAL EQUALS THE FINAL        *  IES  79Ø
C    VALUE OF NZ, THEN ITERATION WAS NOT INVOKED SUCCESSFULLY.          *  IES  8ØØ
C                                                                       *  IES  81Ø
C ************************************************************************IES  82Ø
      DATA CUTOFF /Ø.5DØ/                                                IES  83Ø
C ************************************************************************IES  84Ø
C                                                                       *  IES  85Ø
      DATA RATIO /Ø.Ø625DØ/, ROOTRT /Ø.25DØ/                             IES  86Ø
C                                                                       *  IES  87Ø
C    THESE CONSTANTS DEPEND ON THE NUMERICAL INTEGRATION RULE           *  IES  88Ø
C    BEING USED. SEE THE DISCUSSION OF IESIMP IN ORDER TO SET THEM      *  IES  89Ø
C    PROPERLY WHEN CHANGING NUMERICAL INTEGRATION RULES. RATIO          *  IES  9ØØ
C    DEPENDS ON THE RATE OF CONVERGENCE OF THE RULE BEING USED.         *  IES  91Ø
C    ROOTRT USUALLY EQUALS SQRT(RATIO).                                 *  IES  92Ø
C                                                                       *  IES  93Ø
C ************************************************************************IES  94Ø
C    SET UP RELATIVE BASE ADDRESSES FOR THE VARIOUS ARRAYS INTO WHICH   IES  95Ø
C    W IS TO BE SPLIT.                                                  IES  96Ø
      N = NUPPER                                                         IES  97Ø
```

```
      M = MUPPER                                                        IES  98Ø
      NSQ = N*N                                                         IES  99Ø
      I1 = 1                                                            IES 1ØØØ
      I2 = I1 + NSQ                                                     IES 1Ø1Ø
      I3 = I2 + NSQ                                                     IES 1Ø2Ø
      I4 = I3 + (NSQ+N)/2                                               IES 1Ø3Ø
      I5 = I4 + (NSQ+N)/2                                               IES 1Ø4Ø
      I6 = I5 + M                                                       IES 1Ø5Ø
      I7 = I6 + M                                                       IES 1Ø6Ø
      I8 = I7 + N                                                       IES 1Ø7Ø
      I9 = I8 + N                                                       IES 1Ø8Ø
      I1Ø = I9 + M                                                      IES 1Ø9Ø
      I11 = I1Ø + N                                                     IES 11ØØ
      I12 = I11 + M                                                     IES 111Ø
      I13 = I12 + M                                                     IES 112Ø
      I14 = I13 + M                                                     IES 113Ø
      I15 = I14 + N                                                     IES 114Ø
      I16 = I15 + M                                                     IES 115Ø
      I17 = I16 + M                                                     IES 116Ø
      I18 = I17 + N                                                     IES 117Ø
      I19 = I18 + M                                                     IES 118Ø
      I2Ø = I19 + 4*N                                                   IES 119Ø
      NHALF = (N+1)/2                                                   IES 12ØØ
      CALL IESP(KERNEL, RHFCN, A, B, NZ, EPS, IFLAG, X, NUPPER,         IES 121Ø
     * MUPPER, IFLGR2, IER, RATIO, ROOTRT, CUTOFF, NHALF, W, W(I1),     IES 122Ø
     * W(I2), W(I3), W(I4), W(I5), W(I6), W(I7), W(I8), W(I9),          IES 123Ø
     * W(I1Ø), W(I11), W(I12), W(I13), W(I14), W(I15), W(I16),          IES 124Ø
     * W(I17), W(I18), W(I19), W(I2Ø))                                  IES 125Ø
      RETURN                                                            IES 126Ø
      END                                                               IES 127Ø



      SUBROUTINE IESP(KERNEL, RHFCN, A, B, NZ, EPS, IFLAG, X, NUP,      IES   1Ø
     * MUP, IFLGR2, IER, RATIO, ROOTRT, CUTOFF, NHALF, T, LUFACT,       IES   2Ø
     * KMM, KMN, KNM, RHS, R, RH, DELN, TM, TN, XM, XMZ, WM, WN,        IES   3Ø
     * OLDX, SAVE, XN, SAVE2, ASIDE, ASIDE3)                            IES   4Ø
C THIS ROUTINE CONTROLS THE SOLUTION OF THE INTEGRAL EQUATION.          IES   5Ø
      DOUBLE PRECISION KERNEL, RHFCN, A, B, EPS, X, RATIO, ROOTRT,      IES   6Ø
     * CUTOFF, T, LUFACT, KMM, KMN, KNM, RHS, R, RH, DELN, TM, TN,      IES   7Ø
     * XM, XMZ, WM, WN, OLDX, SAVE, SAVE2, XN, ASIDE, ASIDE2,           IES   8Ø
     * ASIDE3, R2, R1RAT, NUMR2, DENR2, NORM, DMIN1, DSQRT, ERROR,      IES   9Ø
     * TEMP, TEST, R1, DENR1, NUMR1, RT, DET                            IES  1ØØ
      INTEGER OLDM, TWICE, FLAG                                         IES  11Ø
      DIMENSION X(MUP), T(MUP), LUFACT(NUP,NUP), KMM(NUP,NUP),          IES  12Ø
     * KMN(NUP,NHALF), KNM(NHALF,NUP), RHS(MUP), R(MUP), RH(NUP),       IES  13Ø
     * DELN(NUP), TM(MUP), TN(NUP), XM(MUP), XMZ(MUP), WM(MUP),         IES  14Ø
     * WN(NUP), OLDX(MUP), SAVE(MUP), XN(NUP), SAVE2(MUP),              IES  15Ø
     * ASIDE(NUP,4), ASIDE2(5), ASIDE3(NUP,NUP)                         IES  16Ø
      COMMON /INFO/ R1, R2, N                                           IES  17Ø
      EXTERNAL KERNEL, RHFCN                                            IES  18Ø
C *********************************************************************IES  19Ø
C                                                                       *IES  2ØØ
      TWICE(KK) = 2*KK - 1                                              IES  21Ø
C                                                                       *IES  22Ø
C      THIS FUNCTION GIVES THE FORMULA FOR INCREASING THE NUMBER OF     *IES  23Ø
C    NODE POINTS. WITH ANOTHER INTEGRATION RULE, IT MAY BE              *IES  24Ø
C    NECESSARY TO CHANGE THE DEFINITION OF TWICE(KK).                   *IES  25Ø
C                                                                       *IES  26Ø
C *********************************************************************IES  27Ø
C    INITIALIZATION.                                                    IES  28Ø
      IER = Ø                                                           IES  29Ø
      LOOP = 1                                                          IES  3ØØ
      N = NZ                                                            IES  31Ø
      R2 = Ø.5DØ                                                        IES  32Ø
      M = TWICE(N)                                                      IES  33Ø
      R1RAT = ROOTRT                                                    IES  34Ø
C STAGE A. DIRECT SOLUTION OF LINEAR SYSTEM (I-KN)*XN=RHS, WHILE        IES  35Ø
C TRYING TO FIND A GOOD APPROXIMATE INVERSE TO IMPLEMENT THE            IES  36Ø
C ITERATIVE METHOD OF SOLUTION.                                         IES  37Ø
C CREATE NODES TN(I) AND WEIGHTS WN(I), I=1,...,N.                      IES  38Ø
      CALL WANDT(WN, TN, N, A, B)                                       IES  39Ø
C CREATE SYSTEM (I-KN)*XN=RHFCN.                                        IES  4ØØ
```

```
      DO 2Ø I=1,N                                                        IES  41Ø
        DO 1Ø J=1,N                                                      IES  42Ø
          LUFACT(I,J) = -WN(J)*KERNEL(TN(I),TN(J))                       IES  43Ø
   1Ø   CONTINUE                                                         IES  44Ø
        XN(I) = RHFCN(TN(I))                                             IES  45Ø
        LUFACT(I,I) = LUFACT(I,I) + 1.ØDØ                                IES  46Ø
   2Ø CONTINUE                                                           IES  47Ø
      GO TO 6Ø                                                           IES  48Ø
C THIS IS ENTRANCE FOR AN INCREASED VALUE OF N, USING PREVIOUSLY         IES  49Ø
C STORED VALUES IN KMM AND RHS.                                          IES  5ØØ
   3Ø DO 5Ø J=1,N                                                        IES  51Ø
        DO 4Ø I=1,N                                                      IES  52Ø
          LUFACT(I,J) = -KMM(I,J)                                        IES  53Ø
   4Ø   CONTINUE                                                         IES  54Ø
        WN(J) = WM(J)                                                    IES  55Ø
        TN(J) = TM(J)                                                    IES  56Ø
        XN(J) = RHS(J)                                                   IES  57Ø
        LUFACT(J,J) = LUFACT(J,J) + 1.ØDØ                                IES  58Ø
   5Ø CONTINUE                                                           IES  59Ø
C THIS IS THE ENTRANCE WHEN ITERATION IN STAGE B FAILS AND WE NEED        IES  6ØØ
C TO INCREASE N TO OBTAIN A BETTER ITERATIVE RATE.                       IES  61Ø
   6Ø CONTINUE                                                           IES  62Ø
C    SOLVE (I-KN)*XN=RHFCN AT ALL TN(I).ALSO OBTAIN THE LU               IES  63Ø
C    DECOMPOSITION FOR LATER USE IN THE STAGE B ITERATIVE METHOD.        IES  64Ø
C ********************************************************************   IES  65Ø
C                                                                    *   IES  66Ø
      CALL LINSYS(LUFACT, LUFACT, N, XN, XN, 1, DET, NUP)                IES  67Ø
C                                                                    *   IES  68Ø
C    THIS LINEAR EQUATION SOLVER MAY BE REPLACED BY ANOTHER PROGRAM,*    IES  69Ø
C    BUT CARE MUST BE TAKEN THAT THE NEW PROGRAM DOES THE SAME JOB  *    IES  7ØØ
C    AS LINSYS. THE PROGRAM LINSYS IS ALSO REFERENCED IN THE        *    IES  71Ø
C    SUBROUTINE ITERT.                                              *    IES  72Ø
C                                                                    *   IES  73Ø
C ********************************************************************   IES  74Ø
      IF (LOOP.EQ.1) GO TO 11Ø                                           IES  75Ø
      IF (LOOP.EQ.2) GO TO 8Ø                                            IES  76Ø
C COMPUTE THE RATE OF CONVERGENCE, AND COMPARE WITH THE                  IES  77Ø
C THEORTICAL RATIO.                                                      IES  78Ø
      NUMR2 = NORM(XN,OLDX,N,1)                                          IES  79Ø
      R2 = DMIN1(Ø.5DØ,NUMR2/DENR2)                                      IES  8ØØ
      IF ((R2.LT.RATIO) .AND. (IFLGR2.EQ.Ø)) R2 = RATIO                  IES  81Ø
      R1RAT = DMIN1(DSQRT(R2),ROOTRT)                                    IES  82Ø
      IF (IFLGR2.EQ.Ø) R1RAT = ROOTRT                                    IES  83Ø
C CHECK FOR ERROR IN XN USING TEST INVOLVING R2 AND OLDX,ACCORDING       IES  84Ø
C TO THEORY FOR ASYMPTOTIC ERROR BOUNDS.                                 IES  85Ø
   7Ø ERROR = (R2/(1.ØDØ-R2))*NUMR2                                      IES  86Ø
      IF (IFLAG.EQ.1) ERROR = ERROR/NORM(XN,XN,N,Ø)                      IES  87Ø
      IF ((IFLGR2.EQ.1) .AND. (R2.LT.RATIO)) ERROR = 2.Ø*ERROR           IES  88Ø
      IF (ERROR.LE.EPS) GO TO 9Ø                                         IES  89Ø
      DENR2 = NUMR2                                                      IES  9ØØ
      GO TO 11Ø                                                          IES  91Ø
C ENTRANCE FOR LOOP=2.                                                   IES  92Ø
   8Ø NUMR2 = NORM(XN,OLDX,N,1)                                          IES  93Ø
      DENR2 = Ø.ØDØ                                                      IES  94Ø
      GO TO 7Ø                                                           IES  95Ø
C SET UP T,X,EPS,NZ FOR SUCCESSFUL RETURN.                               IES  96Ø
   9Ø DO 1ØØ I=1,N                                                       IES  97Ø
        X(I) = XN(I)                                                     IES  98Ø
        T(I) = TN(I)                                                     IES  99Ø
  1ØØ CONTINUE                                                           IES 1ØØØ
      EPS = ERROR                                                        IES 1Ø1Ø
      NZ = N                                                             IES 1Ø2Ø
      RETURN                                                             IES 1Ø3Ø
C INITIALIZE FOR SOLVING (I-KM)*XM=RHFCN ITERATIVELY.                    IES 1Ø4Ø
  11Ø CALL WANDT(WM, TM, M, A, B)                                        IES 1Ø5Ø
      FLAG = Ø                                                           IES 1Ø6Ø
      CALL INTERP(TM, WM, XMZ, M, TN, WN, XN, N, KERNEL, RHFCN,          IES 1Ø7Ø
     * RHS, KMN, NHALF, NUP)                                            IES 1Ø8Ø
      DO 12Ø I=1,M                                                       IES 1Ø9Ø
        OLDX(I) = XMZ(I)                                                 IES 11ØØ
  12Ø CONTINUE                                                           IES 111Ø
      CALL ITERT(KERNEL, RHFCN, N, TN, WN, M, TM, WM, XM, XMZ, KMM,      IES 112Ø
     * KMN, KNM, RHS, LUFACT, R, RH, DELN, NUP, NHALF, FLAG)            IES 113Ø
      DENR1 = NORM(XM,XMZ,M,1)                                           IES 114Ø
      FLAG = 1                                                           IES 115Ø
      DO 13Ø I=1,M                                                       IES 116Ø
```

```
      XMZ(I) = XM(I)                                                    IES 117Ø
 13Ø CONTINUE                                                           IES 118Ø
     CALL ITERT(KERNEL, RHFCN, N, TN, WN, M, TM, WM, XM, XMZ, KMM,      IES 119Ø
    * KMN, KNM, RHS, LUFACT, R, RH, DELN, NUP, NHALF, FLAG)             IES 12ØØ
     NUMR1 = NORM(XM,XMZ,M,1)                                           IES 121Ø
C CHECK ON THE SPEED OF CONVERGENCE OF ITERATIVE METHOD. IF IT IS        IES 122Ø
C SUFFICIENTLY RAPID, THEN FIX N AND GO TO STAGE B.                     IES 123Ø
     R1 = NUMR1/DENR1                                                   IES 124Ø
     IF (M.GT.NUP) GO TO 19Ø                                            IES 125Ø
     IF (R1.LE.R1RAT) GO TO 15Ø                                         IES 126Ø
C REINITIALIZE FOR SOLVING (I-KN)*XN=RHFCN AGAIN WITH A LARGER N.       IES 127Ø
 14Ø N = M                                                              IES 128Ø
     LOOP = LOOP + 1                                                    IES 129Ø
     M = TWICE(N)                                                       IES 13ØØ
     GO TO 3Ø                                                           IES 131Ø
C SAVE INFORMATION IN CASE STAGE B ABORTS AT A LARGER VALUE OF M         IES 132Ø
C AND STAGE A HAS TO BE RETURNED TO.                                    IES 133Ø
 15Ø DO 16Ø I=1,M                                                       IES 134Ø
       ASIDE(I,1) = OLDX(I)                                             IES 135Ø
       ASIDE(I,2) = WM(I)                                               IES 136Ø
       ASIDE(I,3) = TM(I)                                               IES 137Ø
       ASIDE(I,4) = RHS(I)                                              IES 138Ø
 16Ø CONTINUE                                                           IES 139Ø
     ASIDE2(1) = LOOP                                                   IES 14ØØ
     ASIDE2(2) = M                                                      IES 141Ø
     ASIDE2(3) = R2                                                     IES 142Ø
     ASIDE2(4) = DENR2                                                  IES 143Ø
     ASIDE2(5) = R1RAT                                                  IES 144Ø
     DO 18Ø I=1,M                                                       IES 145Ø
       DO 17Ø J=1,M                                                     IES 146Ø
         ASIDE3(I,J) = KMM(I,J)                                         IES 147Ø
 17Ø   CONTINUE                                                         IES 148Ø
 18Ø CONTINUE                                                           IES 149Ø
     GO TO 24Ø                                                          IES 15ØØ
C STAGE B. ITERATIVE METHOD OF SOLUTION OF (I-KM)*XM=RHS.               IES 151Ø
 19Ø IF (R1.LE.CUTOFF) GO TO 24Ø                                        IES 152Ø
C IF ITERATES ARE CONVERGING VERY SLOWLY OR NOT AT ALL, THEN            IES 153Ø
C RETURN WITHOUT FURTHER ATTEMPTS TO LESSEN ERROR.                      IES 154Ø
 2ØØ IF (LOOP.NE.1) GO TO 23Ø                                           IES 155Ø
     EPS = Ø.ØDØ                                                        IES 156Ø
     IER = 2                                                            IES 157Ø
 21Ø DO 22Ø I=1,N                                                       IES 158Ø
       X(I) = XN(I)                                                     IES 159Ø
       T(I) = TN(I)                                                     IES 16ØØ
 22Ø CONTINUE                                                           IES 161Ø
     NZ = N                                                             IES 162Ø
     RETURN                                                             IES 163Ø
 23Ø EPS = ERROR                                                        IES 164Ø
     IER = 1                                                            IES 165Ø
     GO TO 21Ø                                                          IES 166Ø
C TEST TO SEE IF THE CURRENT ITERATE XM IS SUFFICIENTLY ACCURATE        IES 167Ø
C COMPARED TO THE TRUE XM.                                              IES 168Ø
 24Ø TEMP = NORM(XM,OLDX,M,1)                                           IES 169Ø
     RT = DMIN1(RATIO,R2)                                               IES 17ØØ
     TEST = ((1.ØDØ-R1)/R1)*(RT/(1.ØDØ-RT))*TEMP                       IES 171Ø
     IF (NUMR1.LE.TEST) GO TO 32Ø                                       IES 172Ø
C ITERATE NOT ACCURATE. INITIALIZE FOR COMPUTATION OF ANOTHER           IES 173Ø
C ITERATE.                                                              IES 174Ø
     DENR1 = NUMR1                                                      IES 175Ø
     DO 25Ø I=1,M                                                       IES 176Ø
       XMZ(I) = XM(I)                                                   IES 177Ø
 25Ø CONTINUE                                                           IES 178Ø
     CALL ITERT(KERNEL, RHFCN, N, TN, WN, M, TM, WM, XM, XMZ, KMM,      IES 179Ø
    * KMN, KNM, RHS, LUFACT, R, RH, DELN, NUP, NHALF, FLAG)             IES 18ØØ
     NUMR1 = NORM(XM,XMZ,M,1)                                           IES 181Ø
     R1 = NUMR1/DENR1                                                   IES 182Ø
     IF (R1.LE.CUTOFF) GO TO 24Ø                                        IES 183Ø
C THIS IS ENTRANCE FOR CASE WHERE ITERATION FAILS IN STAGE B.           IES 184Ø
C PARAMETERS MUST BE RESET FOR A RETURN TO STAGE A, OR IF N CANNOT      IES 185Ø
C BE INCREASED, FOR AN ERROR EXIT FROM IESIMP.                          IES 186Ø
 26Ø MNEW = ASIDE2(2)                                                   IES 187Ø
     IF (MNEW.GT.NUP) GO TO 29Ø                                         IES 188Ø
     IF (M.EQ.TWICE(N)) GO TO 14Ø                                       IES 189Ø
     N = MNEW                                                           IES 19ØØ
     DO 28Ø J=1,N                                                       IES 191Ø
       DO 27Ø I=1,N                                                     IES 192Ø
```

```
          LUFACT(I,J) = -ASIDE3(I,J)                                    IES 193Ø
   27Ø    CONTINUE                                                      IES 194Ø
          OLDX(J) = ASIDE(J,1)                                          IES 195Ø
          WN(J) = ASIDE(J,2)                                            IES 196Ø
          TN(J) = ASIDE(J,3)                                            IES 197Ø
          XN(J) = ASIDE(J,4)                                            IES 198Ø
          LUFACT(J,J) = LUFACT(J,J) + 1.ØDØ                             IES 199Ø
   28Ø CONTINUE                                                         IES 2ØØØ
       M = TWICE(N)                                                     IES 2Ø1Ø
       LOOP = ASIDE2(1) + 1.ØDØ                                         IES 2Ø2Ø
       R2 = ASIDE2(3)                                                   IES 2Ø3Ø
       DENR2 = ASIDE2(4)                                                IES 2Ø4Ø
       R1RAT = ASIDE2(5)                                                IES 2Ø5Ø
       GO TO 6Ø                                                         IES 2Ø6Ø
C ABORTIVE EXIT FROM STAGE B. N CANNOT BE INCREASED FURTHER, AND        IES 2Ø7Ø
C R1 IS NOT SUFFICIENTLY SMALL.                                         IES 2Ø8Ø
   29Ø IF (M.EQ.TWICE(N)) GO TO 2ØØ                                     IES 2Ø9Ø
       DO 3ØØ I=1,OLDM                                                  IES 21ØØ
         T(I) = SAVE(I)                                                 IES 211Ø
   3ØØ CONTINUE                                                         IES 212Ø
       NZ = OLDM                                                        IES 213Ø
       IF (LOOP.EQ.1) GO TO 31Ø                                         IES 214Ø
       EPS = ERROR                                                      IES 215Ø
       IER = 1                                                          IES 216Ø
       RETURN                                                           IES 217Ø
   31Ø EPS = Ø.Ø                                                        IES 218Ø
       IER = 2                                                          IES 219Ø
       RETURN                                                           IES 22ØØ
C AN ACCURATE VALUE OF XM HAS BEEN OBTAINED. R2 IS TO BE COMPUTED       IES 221Ø
C AND COMPARED TO RATIO. THEN THE ERROR IN XM IS TO BE ESTIMATED.       IES 222Ø
   32Ø IF (LOOP.EQ.1) GO TO 34Ø                                         IES 223Ø
       NUMR2 = TEMP                                                     IES 224Ø
       R2 = DMIN1(NUMR2/DENR2,Ø.5DØ)                                    IES 225Ø
       IF ((R2.LT.RATIO) .AND. (IFLGR2.EQ.Ø)) R2 = RATIO                IES 226Ø
       DENR2 = NUMR2                                                    IES 227Ø
   33Ø ERROR = (R2/(1.ØDØ-R2))*TEMP                                     IES 228Ø
       IF (IFLAG.EQ.1) ERROR = ERROR/NORM(XM,XM,M,Ø)                    IES 229Ø
       IF ((IFLGR2.EQ.1) .AND. (R2.LT.RATIO)) ERROR = 2.Ø*ERROR         IES 23ØØ
       IF (ERROR.LE.EPS) GO TO 4ØØ                                      IES 231Ø
       MNEW = TWICE(M)                                                  IES 232Ø
       IF (MNEW.LE.MUP) GO TO 35Ø                                       IES 233Ø
       IER = 1                                                          IES 234Ø
       GO TO 4ØØ                                                        IES 235Ø
   34Ø DENR2 = TEMP                                                     IES 236Ø
       LOOP = 2                                                         IES 237Ø
       GO TO 33Ø                                                        IES 238Ø
C ERROR NOT SUFFICIENTLY SMALL. M IS INCREASED AND TWO MORE             IES 239Ø
C ITERATES ARE COMPUTED WITH A NEW M.                                   IES 24ØØ
   35Ø DO 36Ø I=1,M                                                     IES 241Ø
         X(I) = XM(I)                                                   IES 242Ø
   36Ø CONTINUE                                                         IES 243Ø
       OLDM = M                                                         IES 244Ø
       M = MNEW                                                         IES 245Ø
       DO 37Ø I=1,OLDM                                                  IES 246Ø
         SAVE2(I) = WM(I)                                               IES 247Ø
         SAVE(I) = TM(I)                                                IES 248Ø
   37Ø CONTINUE                                                         IES 249Ø
       CALL WANDT(WM, TM, M, A, B)                                      IES 25ØØ
       FLAG = Ø                                                         IES 251Ø
       CALL INTERP(TM, WM, XMZ, M, SAVE, SAVE2, XM, OLDM, KERNEL,       IES 252Ø
      * RHFCN, RHS, KMN, NHALF, NUP)                                    IES 253Ø
       DO 38Ø I=1,M                                                     IES 254Ø
         OLDX(I) = XMZ(I)                                               IES 255Ø
   38Ø CONTINUE                                                         IES 256Ø
       CALL ITERT(KERNEL, RHFCN, N, TN, WN, M, TM, WM, XM, XMZ, KMM,    IES 257Ø
      * KMN, KNM, RHS, LUFACT, R, RH, DELN, NUP, NHALF, FLAG)           IES 258Ø
       DENR1 = NORM(XM,XMZ,M,1)                                         IES 259Ø
       FLAG = 1                                                         IES 26ØØ
       DO 39Ø I=1,M                                                     IES 261Ø
         XMZ(I) = XM(I)                                                 IES 262Ø
   39Ø CONTINUE                                                         IES 263Ø
       CALL ITERT(KERNEL, RHFCN, N, TN, WN, M, TM, WM, XM, XMZ, KMM,    IES 264Ø
      * KMN, KNM, RHS, LUFACT, R, RH, DELN, NUP, NHALF, FLAG)           IES 265Ø
       NUMR1 = NORM(XM,XMZ,M,1)                                         IES 266Ø
       R1 = NUMR1/DENR1                                                 IES 267Ø
       IF (R1.LE.CUTOFF) GO TO 24Ø                                      IES 268Ø
```

```
      GO TO 26Ø                                                    IES 269Ø
  4ØØ DO 41Ø I=1,M                                                 IES 27ØØ
        X(I) = XM(I)                                               IES 271Ø
        T(I) = TM(I)                                               IES 272Ø
  41Ø CONTINUE                                                     IES 273Ø
      NZ = M                                                       IES 274Ø
      EPS = ERROR                                                  IES 275Ø
      RETURN                                                       IES 276Ø
      END                                                          IES 277Ø



      SUBROUTINE WANDT(W, T, N, A, B)                              WAN   1Ø
C THIS ROUTINE CALCULATES THE WEIGHTS AND NODE POINTS FOR          WAN   2Ø
C SIMPSONS RULE ON (A,B), WITH N EVENLY SPACED NODES.              WAN   3Ø
      DOUBLE PRECISION W, T, A, B, FN, FI, TEMP, H                 WAN   4Ø
      DIMENSION W(N), T(N)                                         WAN   5Ø
C CALCULATE NODE POINTS.                                           WAN   6Ø
      FN = N                                                       WAN   7Ø
      H = (B-A)/(FN-1.ØDØ)                                         WAN   8Ø
      DO 1Ø I=1,N                                                  WAN   9Ø
        FI = I                                                     WAN  1ØØ
        T(I) = A + (FI-1.ØDØ)*H                                    WAN  11Ø
   1Ø CONTINUE                                                     WAN  12Ø
C CALCULATE WEIGHTS.                                               WAN  13Ø
      W(1) = H/3.ØDØ                                               WAN  14Ø
      W(N) = H/3.ØDØ                                               WAN  15Ø
      NM1 = N - 1                                                  WAN  16Ø
      TEMP = 4.ØDØ*H/3.ØDØ                                         WAN  17Ø
      DO 2Ø I=2,NM1,2                                              WAN  18Ø
        W(I) = TEMP                                                WAN  19Ø
   2Ø CONTINUE                                                     WAN  2ØØ
      IF (N.EQ.3) RETURN                                           WAN  21Ø
      TEMP = 2.ØDØ*H/3.ØDØ                                         WAN  22Ø
      NM2 = N - 2                                                  WAN  23Ø
      DO 3Ø I=3,NM2,2                                              WAN  24Ø
        W(I) = TEMP                                                WAN  25Ø
   3Ø CONTINUE                                                     WAN  26Ø
      RETURN                                                       WAN  27Ø
      END                                                          WAN  28Ø



      SUBROUTINE INTERP(TM, WM, XM, M, TN, WN, XN, N, KERNEL,      INT   1Ø
     * RHFCN, RHS, KMN, NHALF, NUP)                                INT   2Ø
C USE THE VALUES OF XN(I), I=1,...,N, TO CALGULATE THE NYSTROM     INT   3Ø
C INTERPOLATES XM(I), I=1,...,M.                                   INT   4Ø
      DOUBLE PRECISION KERNEL, RHFCN, TM, WM, XM, TN, WN, XN, RHS, INT   5Ø
     * KMN                                                         INT   6Ø
      DIMENSION TM(M), WM(M), XM(M), TN(N), WN(N), XN(N), RHS(M),  INT   7Ø
     * KMN(NUP,NHALF)                                             INT   8Ø
      IF (M.GT.NUP) GO TO 6Ø                                       INT   9Ø
C SINCE M .LE. NUPPER, SAVE K(TM(I),TN(J))=KMN(I,J) AND            INT  1ØØ
C RHS(I)=RHFCN(TM(I)) FOR LATER USE IN ITERT.                      INT  11Ø
      DO 2Ø I=1,M                                                  INT  12Ø
        DO 1Ø J=1,N                                                INT  13Ø
          KMN(I,J) = WN(J)*KERNEL(TM(I),TN(J))                     INT  14Ø
   1Ø   CONTINUE                                                   INT  15Ø
   2Ø CONTINUE                                                     INT  16Ø
      DO 3Ø I=1,M                                                  INT  17Ø
        RHS(I) = RHFCN(TM(I))                                      INT  18Ø
        XM(I) = RHS(I)                                             INT  19Ø
   3Ø CONTINUE                                                     INT  2ØØ
C CALCULATE NYSTROM INTERPOLATING FORMULA.                         INT  21Ø
      DO 5Ø I=1,M                                                  INT  22Ø
        DO 4Ø J=1,N                                                INT  23Ø
          XM(I) = XM(I) + KMN(I,J)*XN(J)                           INT  24Ø
   4Ø   CONTINUE                                                   INT  25Ø
   5Ø CONTINUE                                                     INT  26Ø
      RETURN                                                       INT  27Ø
C M .GT. NUPPER, SO SAVE JUST RHS(I) FOR LATER USE IN ITERT.       INT  28Ø
C CALCULATE NYSTROM INTERPOLATING FORMULA.                         INT  29Ø
   6Ø DO 8Ø I=1,M                                                  INT  3ØØ
        RHS(I) = RHFCN(TM(I))                                      INT  31Ø
        XM(I) = RHS(I)                                             INT  32Ø
        DO 7Ø J=1,N                                                INT  33Ø
```

```
            XM(I) = XM(I) + WN(J)*KERNEL(TM(I),TN(J))*XN(J)                INT  34Ø
    7Ø    CONTINUE                                                         INT  35Ø
    8Ø CONTINUE                                                            INT  36Ø
       RETURN                                                              INT  37Ø
       END                                                                 INT  38Ø



       SUBROUTINE ITERT(KERNEL, RHFCN, N, TN, WN, M, TM, WM, XM,           ITE   1Ø
      * XMZ, KMM, KMN, KNM, RHS, LUFACT, R, RH, DELN, NUP, NHALF,         ITE   2Ø
      * IFLG)                                                             ITE   3Ø
C THIS ROUTINE CALCULATES ONE ITERATE XM GIVEN THE INITIAL GUESS          ITE   4Ø
C XMZ. THE ROUTINE IS DIVIDED ACCORDING TO WHETHER OR NOT                 ITE   5Ø
C M .GT. NUPPER.                                                          ITE   6Ø
       DOUBLE PRECISION KERNEL, RHFCN, TN, WN, TM, WM, XM, XMZ, KMM,      ITE   7Ø
      * KMN, KNM, RHS, LUFACT, R, RH, DELN, SUM, DET                     ITE   8Ø
       DIMENSION TN(N), WN(N), TM(M), WM(M), XM(M), XMZ(M),               ITE   9Ø
      * KMM(NUP,NUP), KMN(NUP,NHALF), KNM(NHALF,NUP), RHS(M),            ITE  1ØØ
      * LUFACT(NUP,NUP), R(M), RH(N), DELN(N)                            ITE  11Ø
C M .GT. NUPPER MEANS THAT THE MATRICES KMM,KMN,KNM CAN NO LONGER         ITE  12Ø
C BE STORED DUE TO LACK OF SPACE.                                         ITE  13Ø
       IF (M.GT.NUP) GO TO 12Ø                                            ITE  14Ø
       IF (IFLG.EQ.1) GO TO 4Ø                                            ITE  15Ø
C IF IFLG=Ø, THEN THE MATRICES KMM AND KNM MUST BE COMPUTED               ITE  16Ø
C AND STORED.                                                             ITE  17Ø
       DO 3Ø J=1,M                                                        ITE  18Ø
         DO 1Ø I=1,M                                                      ITE  19Ø
           KMM(I,J) = WM(J)*KERNEL(TM(I),TM(J))                           ITE  2ØØ
    1Ø   CONTINUE                                                         ITE  21Ø
         DO 2Ø I=1,N                                                      ITE  22Ø
           KNM(I,J) = WM(J)*KERNEL(TN(I),TM(J))                           ITE  23Ø
    2Ø   CONTINUE                                                         ITE  24Ø
    3Ø CONTINUE                                                           ITE  25Ø
C COMPUTE  RESIDUALS R(I)=RHFCN(TM(I))-XMZ(I)+KM(TM(I))*XMZ(I)            ITE  26Ø
    4Ø DO 6Ø I=1,M                                                        ITE  27Ø
         SUM = Ø.ØDØ                                                      ITE  28Ø
         DO 5Ø J=1,M                                                      ITE  29Ø
           SUM = SUM + KMM(I,J)*XMZ(J)                                    ITE  3ØØ
    5Ø   CONTINUE                                                         ITE  31Ø
         R(I) = RHS(I) - (XMZ(I)-SUM)                                     ITE  32Ø
    6Ø CONTINUE                                                           ITE  33Ø
C COMPUTE RH=KM*R AT ALL TN(I).                                           ITE  34Ø
       DO 8Ø I=1,N                                                        ITE  35Ø
         RH(I) = Ø.ØDØ                                                    ITE  36Ø
         DO 7Ø J=1,M                                                      ITE  37Ø
           RH(I) = RH(I) + KNM(I,J)*R(J)                                  ITE  38Ø
    7Ø   CONTINUE                                                         ITE  39Ø
    8Ø CONTINUE                                                           ITE  4ØØ
C    CALCULATE DELN=((I-KN)**(-1))*KM*R AT ALL TN(I).                     ITE  41Ø
C *********************************************************************  ITE  42Ø
C                                                                     *   ITE  43Ø
       CALL LINSYS(LUFACT, LUFACT, N, RH, DELN, 3, DET, NUP)              ITE  44Ø
C                                                                     *   ITE  45Ø
C    SEE THE ORIGINAL REFERENCE IN IESP.                              *   ITE  46Ø
C                                                                     *   ITE  47Ø
C *********************************************************************  ITE  48Ø
C    CALCULATE NEW XM.                                                    ITE  49Ø
       DO 11Ø I=1,M                                                       ITE  5ØØ
         SUM = Ø.ØDØ                                                      ITE  51Ø
         DO 9Ø J=1,M                                                      ITE  52Ø
           SUM = SUM + KMM(I,J)*R(J)                                      ITE  53Ø
    9Ø   CONTINUE                                                         ITE  54Ø
         DO 1ØØ J=1,N                                                     ITE  55Ø
           SUM = SUM + KMN(I,J)*DELN(J)                                   ITE  56Ø
   1ØØ   CONTINUE                                                         ITE  57Ø
         XM(I) = SUM + R(I) + XMZ(I)                                      ITE  58Ø
   11Ø CONTINUE                                                           ITE  59Ø
       RETURN                                                             ITE  6ØØ
C ENTRANCE WHEN M .GT. NUP.                                               ITE  61Ø
C CALCULATE RESIDUALS.                                                    ITE  62Ø
   12Ø DO 14Ø I=1,M                                                       ITE  63Ø
         SUM = Ø.ØDØ                                                      ITE  64Ø
         DO 13Ø J=1,M                                                     ITE  65Ø
           SUM = SUM + WM(J)*KERNEL(TM(I),TM(J))*XMZ(J)                   ITE  66Ø
   13Ø   CONTINUE                                                         ITE  67Ø
         R(I) = RHS(I) - (XMZ(I)-SUM)                                     ITE  68Ø
```

```
  14Ø CONTINUE                                                          ITE  69Ø
C CALCULATE RH=KM*R.                                                    ITE  7ØØ
      DO 16Ø I=1,N                                                      ITE  71Ø
        RH(I) = Ø.ØDØ                                                   ITE  72Ø
        DO 15Ø J=1,M                                                    ITE  73Ø
          RH(I) = RH(I) + WM(J)*KERNEL(TN(I),TM(J))*R(J)                ITE  74Ø
  15Ø   CONTINUE                                                        ITE  75Ø
  16Ø CONTINUE                                                          ITE  76Ø
C ***********************************************************************ITE  77Ø
C                                                                       *ITE  78Ø
      CALL LINSYS(LUFACT, LUFACT, N, RH, DELN, 3, DET, NUP)             ITE  79Ø
C                                                                       *ITE  8ØØ
C    SEE THE ORIGINAL REFERENCE IN IESP.                                *ITE  81Ø
C                                                                       *ITE  82Ø
C ***********************************************************************ITE  83Ø
C    CALCULATE XM.                                                      ITE  84Ø
      DO 19Ø I=1,M                                                      ITE  85Ø
        SUM = Ø.ØDØ                                                     ITE  86Ø
        DO 17Ø J=1,M                                                    ITE  87Ø
          SUM = SUM + WM(J)*KERNEL(TM(I),TM(J))*R(J)                    ITE  88Ø
  17Ø   CONTINUE                                                        ITE  89Ø
        DO 18Ø J=1,N                                                    ITE  9ØØ
          SUM = SUM + WN(J)*KERNEL(TM(I),TN(J))*DELN(J)                 ITE  91Ø
  18Ø   CONTINUE                                                        ITE  92Ø
        XM(I) = SUM + R(I) + XMZ(I)                                     ITE  93Ø
  19Ø CONTINUE                                                          ITE  94Ø
      RETURN                                                            ITE  95Ø
      END                                                               ITE  96Ø
```

```
      DOUBLE PRECISION FUNCTION NORM(X, Y, N, IFLAG)                    NOR   1Ø
C IFLAG=Ø    CALCULATE THE MAXIMUM NORM OF X.                           NOR   2Ø
C IFLAG=1    CALCULATE THE MAXIMUM NORM OF X-Y.                         NOR   3Ø
      DOUBLE PRECISION X, Y, DMAX1, DABS                                NOR   4Ø
      DIMENSION X(N), Y(N)                                              NOR   5Ø
      IF (IFLAG.EQ.1) GO TO 2Ø                                          NOR   6Ø
C FIND THE NORM OF X.                                                   NOR   7Ø
      NORM = Ø.ØDØ                                                      NOR   8Ø
      DO 1Ø I=1,N                                                       NOR   9Ø
        NORM = DMAX1(NORM,DABS(X(I)))                                   NOR  1ØØ
   1Ø CONTINUE                                                          NOR  11Ø
      RETURN                                                            NOR  12Ø
C FIND THE NORM OF X-Y.                                                 NOR  13Ø
   2Ø NORM = Ø.ØDØ                                                      NOR  14Ø
      DO 3Ø I=1,N                                                       NOR  15Ø
        NORM = DMAX1(NORM,DABS(X(I)-Y(I)))                              NOR  16Ø
   3Ø CONTINUE                                                          NOR  17Ø
      RETURN                                                            NOR  18Ø
      END                                                               NOR  19Ø
```

```
      SUBROUTINE LINSYS(A, D, N, B, X, OPTION, DET, MACHIN)             LIN   1Ø
C SOLVE A*X=B, ORDER(A)=N, DIMENSION OF A=MACHIN.                       LIN   2Ø
C OPTION=1   SOLVE A*X=B, LEAVE THE LU DECOMPOSITION A=L*U IN D         LIN   3Ø
C            AND THE PIVOTS IN PIVOT. THE ANSWERS ARE LEFT IN X.        LIN   4Ø
C            IT IS PERMISSABLE TO LET B=X AND D=A, BUT THEN THE         LIN   5Ø
C            ORIGINAL CONTENTS OF A AND B ARE LOST.                     LIN   6Ø
C OPTION=2   CALCULATE DECOMPOSITION A=L*U INCLUDING PIVOTS. SOLVE      LIN   7Ø
C            A*X=B, AND THEN CALCULATE THE RESIDUAL AND ONE             LIN   8Ø
C            CORRECTION. THE CORRECTIONS ARE LEFT IN R, THE NEW         LIN   9Ø
C            VALUE OF X1 IN X, THE RELATIVE ERROR                       LIN  1ØØ
C                     NORM(XØ-X1)/NORM(X1)                              LIN  11Ø
C            IN THE VARIABLE ERROR, AND THE RELATIVE RESIDUAL           LIN  12Ø
C                     NORM(RESIDUAL)/NORM(B)                            LIN  13Ø
C            IN THE VARIABLE RELRSD. THESE VALUES CAN BE OBTAINED       LIN  14Ø
C            USING THE COMMON/LINEAR/ GIVEN BELOW.                      LIN  15Ø
C OPTION=3   SAME AS OPTION=1, EXCEPT A=L*U IS KNOWN AND STORED         LIN  16Ø
C            IN D.                                                      LIN  17Ø
C OPTION=4   SAME AS OPTION=2, EXCEPT A=L*U IS KNOWN AND STORED         LIN  18Ø
C            IN D.                                                      LIN  19Ø
C THE DECOMPOSITION OF A INTO L*U USES SCALED PARTIAL PIVOTING IN       LIN  2ØØ
C THE COLUMNS. FOR OPTIONS 1 AND 2, THE DETERMINANT OF A IS             LIN  21Ø
C CALCULATED AND STORED IN DET. IF DET=Ø, THEN THE ANSWERS ARE          LIN  22Ø
C NONSENSE, D AND X DO NOT CONTAIN USEFUL INFORMATION, AND AND A        LIN  23Ø
```

```
C AND B ARE LEFT UNDISTURBED (UNLESS D=A OR X=B).                    LIN  24Ø
C LINSYS IS PRESENTLY LIMITED TO N .LE. 1ØØ. TO REMOVE THIS          LIN  25Ø
C RESTRICTION,CHANGE THE DIMENSION STATEMENT FOR THE ARRAYS R,       LIN  26Ø
C SCALE, AND PIVOT, WHICH ARE GIVEN IN COMMON/LINEAR/.               LIN  27Ø
      INTEGER OPTION, PIVOT                                          LIN  28Ø
      DOUBLE PRECISION NORMX, NORME, NORMB, NORMR, A, D, B, X, R,    LIN  29Ø
     * SCALE, ERROR, RELRSD, DET, C, TEMP, DMAX1, DABS, SUM          LIN  3ØØ
      DIMENSION A(MACHIN,MACHIN), D(MACHIN,MACHIN), B(N), X(N)       LIN  31Ø
      COMMON /LINEAR/ R(1ØØ), SCALE(1ØØ), ERROR, RELRSD, PIVOT(1ØØ) LIN  32Ø
      ISWIT = 1                                                      LIN  33Ø
      IF (OPTION.GT.2) GO TO 11Ø                                     LIN  34Ø
C PRODUCE LU DECOMPOSITION OF A AND DET(A)                           LIN  35Ø
      DET = Ø.ØDØ                                                    LIN  36Ø
      DO 2Ø I=1,N                                                    LIN  37Ø
        DO 1Ø J=1,N                                                  LIN  38Ø
          D(I,J) = A(I,J)                                            LIN  39Ø
   1Ø   CONTINUE                                                     LIN  4ØØ
   2Ø CONTINUE                                                       LIN  41Ø
C PRODUCE SCALING FACTORS FOR SCALED PARTIAL PIVOTING.               LIN  42Ø
      DO 4Ø I=1,N                                                    LIN  43Ø
        SCALE(I) = Ø.ØDØ                                             LIN  44Ø
        DO 3Ø J=1,N                                                  LIN  45Ø
          SCALE(I) = DMAX1(SCALE(I),DABS(D(I,J)))                    LIN  46Ø
   3Ø   CONTINUE                                                     LIN  47Ø
        IF (SCALE(I).EQ.Ø.ØDØ) RETURN                                LIN  48Ø
   4Ø CONTINUE                                                       LIN  49Ø
      DET = 1.ØDØ                                                    LIN  5ØØ
      NM1 = N - 1                                                    LIN  51Ø
C BEGIN GAUSSIAN ELIMINATION.                                        LIN  52Ø
      DO 1ØØ K=1,NM1                                                 LIN  53Ø
C SELECT PIVOT ROW.                                                  LIN  54Ø
        C = DABS(D(K,K))/SCALE(K)                                    LIN  55Ø
        INDEX = K                                                    LIN  56Ø
        KP1 = K + 1                                                  LIN  57Ø
        DO 5Ø I=KP1,N                                                LIN  58Ø
          TEMP = DABS(D(I,K))/SCALE(I)                               LIN  59Ø
          IF (TEMP.LE.C) GO TO 5Ø                                    LIN  6ØØ
          INDEX = I                                                  LIN  61Ø
          C = TEMP                                                   LIN  62Ø
   5Ø   CONTINUE                                                     LIN  63Ø
        PIVOT(K) = INDEX                                             LIN  64Ø
        IF (INDEX.EQ.K) GO TO 7Ø                                     LIN  65Ø
C SWITCH ROWS OF D.                                                  LIN  66Ø
        DO 6Ø J=K,N                                                  LIN  67Ø
          TEMP = D(K,J)                                              LIN  68Ø
          D(K,J) = D(INDEX,J)                                        LIN  69Ø
          D(INDEX,J) = TEMP                                          LIN  7ØØ
   6Ø   CONTINUE                                                     LIN  71Ø
        TEMP = SCALE(K)                                              LIN  72Ø
        SCALE(K) = SCALE(INDEX)                                      LIN  73Ø
        SCALE(INDEX) = TEMP                                          LIN  74Ø
        DET = -DET                                                   LIN  75Ø
   7Ø   DET = DET*D(K,K)                                             LIN  76Ø
C ELIMINATE UNKNOWN =K FROM BELOW DIAGONAL IN COLUMN K.              LIN  77Ø
        IF (DET.EQ.Ø.ØDØ) RETURN                                     LIN  78Ø
        DO 9Ø I=KP1,N                                                LIN  79Ø
          D(I,K) = D(I,K)/D(K,K)                                     LIN  8ØØ
          TEMP = D(I,K)                                              LIN  81Ø
          DO 8Ø J=KP1,N                                              LIN  82Ø
            D(I,J) = D(I,J) - TEMP*D(K,J)                            LIN  83Ø
   8Ø     CONTINUE                                                   LIN  84Ø
   9Ø   CONTINUE                                                     LIN  85Ø
  1ØØ CONTINUE                                                       LIN  86Ø
      DET = DET*D(N,N)                                               LIN  87Ø
      IF (DET.EQ.Ø.ØDØ) RETURN                                       LIN  88Ø
C THE DECOMPOSITION A=P*L*U IS COMPLETE.                             LIN  89Ø
C BEGIN SOLUTION OF LINEAR SYSTEM P*L*U*X=B                          LIN  9ØØ
C SET R AND X FOR SOLUTION OF A*X=B.                                 LIN  91Ø
  11Ø DO 12Ø I=1,N                                                   LIN  92Ø
        R(I) = B(I)                                                  LIN  93Ø
        X(I) = Ø.ØDØ                                                 LIN  94Ø
  12Ø CONTINUE                                                       LIN  95Ø
      GO TO 17Ø                                                      LIN  96Ø
C SET R=B-A*X FOR COMPUTATION OF CORRECTION.                         LIN  97Ø
  13Ø DO 15Ø I=1,N                                                   LIN  98Ø
        SUM = Ø.ØDØ                                                  LIN  99Ø
```

```
      DO 14Ø J=1,N                                                    LIN 1ØØØ
        SUM = SUM + A(I,J)*X(J)                                       LIN 1Ø1Ø
14Ø   CONTINUE                                                        LIN 1Ø2Ø
      R(I) = B(I) - SUM                                               LIN 1Ø3Ø
15Ø CONTINUE                                                          LIN 1Ø4Ø
    NORMB = Ø.ØDØ                                                     LIN 1Ø5Ø
    NORMR = Ø.ØDØ                                                     LIN 1Ø6Ø
    DO 16Ø I=1,N                                                      LIN 1Ø7Ø
      NORMB = DMAX1(NORMB,DABS(B(I)))                                 LIN 1Ø8Ø
      NORMR = DMAX1(NORMR,DABS(R(I)))                                 LIN 1Ø9Ø
16Ø CONTINUE                                                          LIN 11ØØ
    RELRSD = NORMR/NORMB                                              LIN 111Ø
    ISWIT = 2                                                         LIN 112Ø
C SOLVE P*L*Z=R AND STORE IN R.                                       LIN 113Ø
17Ø NM1 = N - 1                                                       LIN 114Ø
    DO 2ØØ K=1,NM1                                                    LIN 115Ø
      IF (PIVOT(K).EQ.K) GO TO 18Ø                                    LIN 116Ø
      KPIV = PIVOT(K)                                                 LIN 117Ø
      TEMP = R(K)                                                     LIN 118Ø
      R(K) = R(KPIV)                                                  LIN 119Ø
      R(KPIV) = TEMP                                                  LIN 12ØØ
18Ø   KP1 = K + 1                                                     LIN 121Ø
      DO 19Ø I=KP1,N                                                  LIN 122Ø
        R(I) = R(I) - D(I,K)*R(K)                                     LIN 123Ø
19Ø   CONTINUE                                                        LIN 124Ø
2ØØ CONTINUE                                                          LIN 125Ø
C SOLVE U*E=R AND STORE IN R. ALSO LET X=X+E.                         LIN 126Ø
    R(N) = R(N)/D(N,N)                                                LIN 127Ø
    GO TO (21Ø, 22Ø, 21Ø, 22Ø), OPTION                                LIN 128Ø
21Ø X(N) = R(N)                                                       LIN 129Ø
    GO TO 23Ø                                                         LIN 13ØØ
22Ø X(N) = X(N) + R(N)                                                LIN 131Ø
23Ø DO 27Ø II=1,NM1                                                   LIN 132Ø
      I = N - II                                                      LIN 133Ø
      SUM = Ø.ØDØ                                                     LIN 134Ø
      IP1 = I + 1                                                     LIN 135Ø
      DO 24Ø J=IP1,N                                                  LIN 136Ø
        SUM = SUM + D(I,J)*R(J)                                       LIN 137Ø
24Ø   CONTINUE                                                        LIN 138Ø
      R(I) = (R(I)-SUM)/D(I,I)                                        LIN 139Ø
      GO TO (25Ø, 26Ø, 25Ø, 26Ø), OPTION                              LIN 14ØØ
25Ø   X(I) = R(I)                                                     LIN 141Ø
      GO TO 27Ø                                                       LIN 142Ø
26Ø   X(I) = X(I) + R(I)                                              LIN 143Ø
27Ø CONTINUE                                                          LIN 144Ø
C SOLUTION OF LINEAR SYSTEM IS COMPLETE. RETURN FOR OPTION=1,3.       LIN 145Ø
    GO TO (28Ø, 29Ø, 28Ø, 29Ø), OPTION                                LIN 146Ø
28Ø RETURN                                                            LIN 147Ø
29Ø GO TO (13Ø, 3ØØ), ISWIT                                           LIN 148Ø
C CALCULATE ERRORS BASED ON CORRECTION.                               LIN 149Ø
3ØØ NORMX = Ø.ØDØ                                                     LIN 15ØØ
    NORME = Ø.ØDØ                                                     LIN 151Ø
    DO 31Ø I=1,N                                                      LIN 152Ø
      NORMX = DMAX1(NORMX,DABS(X(I)))                                 LIN 153Ø
      NORME = DMAX1(NORME,DABS(R(I)))                                 LIN 154Ø
31Ø CONTINUE                                                          LIN 155Ø
    ERROR = NORME/NORMX                                               LIN 156Ø
    RETURN                                                            LIN 157Ø
    END                                                               LIN 158Ø
```

```
    SUBROUTINE IEGAUS(KERNEL, RHFCN, A, B, EP, IFLAG, X, T, NT,       IEG  1Ø
   * NUPPER, MUPPER, W, IER)                                          IEG  2Ø
C THE INTEGRAL EQUATION BEING SOLVED IS                               IEG  3Ø
C                    B                                                IEG  4Ø
C        X(S) -  INT  KERNEL(S,T)*X(T)*DT = RHFCN(S)                  IEG  5Ø
C                    A                                                IEG  6Ø
C THE METHOD BEING USED IS BASED ON THE NYSTROM METHOD WITH           IEG  7Ø
C GAUSSIAN QUADRATURE, WITH AN ITERATIVE TECHNIQUE OF SOLUTION        IEG  8Ø
C FOR THE RESULTING LINEAR SYSTEM.                                    IEG  9Ø
C KERNEL     THESE ARE DOUBLE PRECISION FUNCTIONS OF TWO AND ONE      IEG  1ØØ
C RHFCN      VARIABLES, RESPECTIVELY. THEY MUST BE DECLARED IN AN     IEG  11Ø
C            EXTERNAL STATEMENT IN THE PROGRAM CALLING IEGAUS.        IEG  12Ø
C EP         THE DESIRED ERROR. THE VARIABLE EP IS CHANGED ON         IEG  13Ø
C            COMPLETION OF THE PROGRAM. SEE THE DISCUSSION OF IER     IEG  14Ø
```

```
C            AND IFLAG FOR MORE INFORMATION.                              IEG  15Ø
C IFLAG =Ø   EP IS INTERPRETED AS AN ABSOLUTE ERROR TOLERANCE.            IEG  16Ø
C       =1   EP IS INTERPRETED AS A RELATIVE ERROR TOLERANCE.             IEG  17Ø
C X          THE COMPUTED APPROXIMATE SOLUTION OF THE INTEGRAL            IEG  18Ø
C            EQUATION, EVALUATED AT THE NODE POINTS IN T, IS              IEG  19Ø
C            STORED IN X ON COMPLETION OF THE ROUTINE. THIS IS            IEG  2ØØ
C            TRUE IRREGARDLESS OF WHETHER OR NOT THE DESIRED ERROR        IEG  21Ø
C            TOLERANCE WAS ATTAINED.                                      IEG  22Ø
C T          CONTAINS THE NODE POINTS AT WHICH THE SOLUTION OF THE        IEG  23Ø
C            INTEGRAL EQUATION IS DESIRED. SEE THE VARIABLE NT FOR        IEG  24Ø
C            MORE INFORMATION.                                            IEG  25Ø
C NT         IF NT=Ø, THEN T AND X WILL BE SET EQUAL TO THE FINAL         IEG  26Ø
C            GAUSSIAN NODES AND THE CORRESPONDING SOLUTION VALUES,        IEG  27Ø
C            AND NT WILL BE SET TO THE NUMBER OF THE SOLUTION             IEG  28Ø
C            VALUES STORED IN X AND T. THE ARRAYS T AND X SHOULD          IEG  29Ø
C            HAVE DIMENSION AT LEAST MUPPER, ASSIGNED IN THE              IEG  3ØØ
C            CALLING PROGRAM.                                             IEG  31Ø
C            IF NT .GT. Ø, THEN T CONTAINS NT USER SUPPLIED NODES         IEG  32Ø
C            AT WHICH THE SOLUTION X IS DESIRED.                          IEG  33Ø
C NUPPER     AN UPPER LIMIT ON THE VARIABLE N IN THIS PROGRAM.            IEG  34Ø
C            N IS THE ORDER OF A LINEAR SYSTEM WHICH IS BEING             IEG  35Ø
C            USED TO ITERATIVELY SOLVE A LARGER LINEAR SYSTEM OF          IEG  36Ø
C            ORDER M WHICH APPROXIMATES THE ABOVE INTEGRAL                IEG  37Ø
C            EQUATION.                                                    IEG  38Ø
C MUPPER     AN UPPER LIMIT ON THE VARIABLE M IN THE PROGRAM.             IEG  39Ø
C            N AND M ARE ALWAYS POWERS OF TWO.                            IEG  4ØØ
C W          TEMPORARY WORKING STORAGE FOR THE PROGRAM. IT MUST           IEG  41Ø
C            CONTAIN AT LEAST 5*NU*NU+9*(NU+MU) POSITIONS, WITH           IEG  42Ø
C            NU=NUPPER, MU=MUPPER.                                        IEG  43Ø
C IER =Ø     THIS ERROR COMPLETION CODE MEANS THE ROUTINE WAS              IEG  44Ø
C            COMPLETED SATISFACTORILY. EP CONTAINS THE PREDICTED          IEG  45Ø
C            ERROR.                                                       IEG  46Ø
C     =1     THE ERROR TEST WAS NOT SATISFIED. EP CONTAINS THE            IEG  47Ø
C            PREDICTED ERROR.                                             IEG  48Ø
C     =2     THE ERROR TEST WAS NOT SATISFIED. EP HAS BEEN SET            IEG  49Ø
C            TO ZERO.                                                     IEG  5ØØ
C     =3     THE ORIGINAL VALUE OF EP WAS TOO SMALL, DUE TO               IEG  51Ø
C            POSSIBLE ILL-CONDITIONING PROBLEMS IN THE INTEGRAL           IEG  52Ø
C            EQUATION. THE VALUE OF EP WAS RESET TO A MORE                IEG  53Ø
C            REALISTIC VALUE, AND THAT TOLERANCE WAS ATTAINED.            IEG  54Ø
C     =4     THE ERROR WAS SATISFACTORY AT THE GAUSSIAN NODE               IEG  55Ø
C            POINTS (IER=Ø), BUT THE INTERPOLATION PROCESS(DUE TO         IEG  56Ø
C            NT .GT. Ø) MAY NOT PRESERVE THIS ACCURACY. CHECK THE         IEG  57Ø
C            VALUE OF NORM(K) FOR POSSIBLE INDICATIONS THAT THE           IEG  58Ø
C            INTEGRAL EQUATION MAY BE ALMOST FIRST KIND. SUCH             IEG  59Ø
C            EQUATIONS ARE QUITE ILL-CONDITIONED. THE ERROR IN EP         IEG  6ØØ
C            IS THE PREDICTED ERROR FOR THE SOLUTION AT THE               IEG  61Ø
C            GAUSSIAN NODE POINTS OF ORDER MFINAL.                        IEG  62Ø
C     =5     THE ANALOGUE OF IER=4, BUT WITH IER=1 AT THE                 IEG  63Ø
C            GAUSSIAN NODE POINTS.                                        IEG  64Ø
C     =6     THE ANALOGUE OF IER=4, BUT WITH IER=3 AT THE                 IEG  65Ø
C            GAUSSIAN NODE POINTS.                                        IEG  66Ø
C IEGAUS IS PRESENTLY LIMITED TO NUPPER .LE. 1ØØ. THIS IS ENTIRELY        IEG  67Ø
C DUE TO A RESTRICTION IN THE PROGRAM LINSYS. TO REMOVE THIS              IEG  68Ø
C RESTRICTION, CHANGE THE DIMENSION STATEMENTS IN THE                     IEG  69Ø
C COMMON/LINEAR/ STATEMENTS IN LINSYS AND THE SUBPROGRAM IEGS.            IEG  7ØØ
      DOUBLE PRECISION KERNEL, RHFCN, A, B, EP, X, T, W, CUTOFF,          IEG  71Ø
     * ROOTRT, UNITRD, R1, R2, FINLEP, NORMK                             IEG  72Ø
      DIMENSION X(1), T(1), W(1)                                          IEG  73Ø
      EXTERNAL KERNEL, RHFCN                                              IEG  74Ø
C *********************************************************************  IEG  75Ø
C                                                                     *   IEG  76Ø
      COMMON /INFO/ R1, R2, FINLEP, NORMK, NFINAL, MFINAL                 IEG  77Ø
C                                                                     *   IEG  78Ø
C    THE NUMBERS IN INFO GIVE ADDITIONAL INFORMATION ABOUT THE        *   IEG  79Ø
C    FUNCTIONING OF IEGAUS. R1 IS THE ITERATIVE RATE OF CONVERGENCE   *   IEG  8ØØ
C    IN THE MOST RECENTLY COMPUTED LINEAR SYSTEM. R2 IS THE RATE OF   *   IEG  81Ø
C    CONVERGENCE OF THE GAUSSIAN QUADRATURE VARIANT OF THE NYSTROM    *   IEG  82Ø
C    METHOD. FINLEP IS THE FINAL VALUE OF EP USED AS THE DESIRED      *   IEG  83Ø
C    ERROR TOLERANCE. USUALLY FINLEP WILL EQUAL THE INPUT VALUE OF    *   IEG  84Ø
C    EP, UNLESS EP WAS MUCH TOO SMALL. NORMK IS AN APPROXIMATE        *   IEG  85Ø
C    VALUE FOR THE NORM OF THE INTEGRAL OPERATOR K, AND IT IS         *   IEG  86Ø
C    CALCULATED ONLY IF NT .GT. Ø.                                    *   IEG  87Ø
C    NFINAL AND MFINAL ARE THE FINAL VALUES OF N AND M USED IN        *   IEG  88Ø
C    IEGAUS. IF NFINAL=MFINAL, THEN ITERATION WAS NOT INVOKED         *   IEG  89Ø
```

```
      DO 14Ø J=1,N                                                    LIN 1ØØØ
        SUM = SUM + A(I,J)*X(J)                                       LIN 1Ø1Ø
 14Ø  CONTINUE                                                        LIN 1Ø2Ø
      R(I) = B(I) - SUM                                               LIN 1Ø3Ø
 15Ø CONTINUE                                                         LIN 1Ø4Ø
     NORMB = Ø.ØDØ                                                    LIN 1Ø5Ø
     NORMR = Ø.ØDØ                                                    LIN 1Ø6Ø
     DO 16Ø I=1,N                                                     LIN 1Ø7Ø
       NORMB = DMAX1(NORMB,DABS(B(I)))                                LIN 1Ø8Ø
       NORMR = DMAX1(NORMR,DABS(R(I)))                                LIN 1Ø9Ø
 16Ø CONTINUE                                                         LIN 11ØØ
     RELRSD = NORMR/NORMB                                             LIN 111Ø
     ISWIT = 2                                                        LIN 112Ø
C SOLVE P*L*Z=R AND STORE IN R.                                       LIN 113Ø
 17Ø NM1 = N - 1                                                      LIN 114Ø
     DO 2ØØ K=1,NM1                                                   LIN 115Ø
       IF (PIVOT(K).EQ.K) GO TO 18Ø                                   LIN 116Ø
       KPIV = PIVOT(K)                                                LIN 117Ø
       TEMP = R(K)                                                    LIN 118Ø
       R(K) = R(KPIV)                                                 LIN 119Ø
       R(KPIV) = TEMP                                                 LIN 12ØØ
 18Ø   KP1 = K + 1                                                    LIN 121Ø
       DO 19Ø I=KP1,N                                                 LIN 122Ø
         R(I) = R(I) - D(I,K)*R(K)                                    LIN 123Ø
 19Ø   CONTINUE                                                       LIN 124Ø
 2ØØ CONTINUE                                                         LIN 125Ø
C SOLVE U*E=R AND STORE IN R. ALSO LET X=X+E.                         LIN 126Ø
     R(N) = R(N)/D(N,N)                                               LIN 127Ø
     GO TO (21Ø, 22Ø, 21Ø, 22Ø), OPTION                               LIN 128Ø
 21Ø X(N) = R(N)                                                      LIN 129Ø
     GO TO 23Ø                                                        LIN 13ØØ
 22Ø X(N) = X(N) + R(N)                                               LIN 131Ø
 23Ø DO 27Ø II=1,NM1                                                  LIN 132Ø
       I = N - II                                                     LIN 133Ø
       SUM = Ø.ØDØ                                                    LIN 134Ø
       IP1 = I + 1                                                    LIN 135Ø
       DO 24Ø J=IP1,N                                                 LIN 136Ø
         SUM = SUM + D(I,J)*R(J)                                      LIN 137Ø
 24Ø   CONTINUE                                                       LIN 138Ø
       R(I) = (R(I)-SUM)/D(I,I)                                       LIN 139Ø
       GO TO (25Ø, 26Ø, 25Ø, 26Ø), OPTION                             LIN 14ØØ
 25Ø   X(I) = R(I)                                                    LIN 141Ø
       GO TO 27Ø                                                      LIN 142Ø
 26Ø   X(I) = X(I) + R(I)                                             LIN 143Ø
 27Ø CONTINUE                                                         LIN 144Ø
C SOLUTION OF LINEAR SYSTEM IS COMPLETE. RETURN FOR OPTION=1,3.       LIN 145Ø
     GO TO (28Ø, 29Ø, 28Ø, 29Ø), OPTION                               LIN 146Ø
 28Ø RETURN                                                           LIN 147Ø
 29Ø GO TO (13Ø, 3ØØ), ISWIT                                          LIN 148Ø
C CALCULATE ERRORS BASED ON CORRECTION.                               LIN 149Ø
 3ØØ NORMX = Ø.ØDØ                                                    LIN 15ØØ
     NORME = Ø.ØDØ                                                    LIN 151Ø
     DO 31Ø I=1,N                                                     LIN 152Ø
       NORMX = DMAX1(NORMX,DABS(X(I)))                                LIN 153Ø
       NORME = DMAX1(NORME,DABS(R(I)))                                LIN 154Ø
 31Ø CONTINUE                                                         LIN 155Ø
     ERROR = NORME/NORMX                                              LIN 156Ø
     RETURN                                                           LIN 157Ø
     END                                                              LIN 158Ø


     SUBROUTINE IEGAUS(KERNEL, RHFCN, A, B, EP, IFLAG, X, T, NT,      IEG  1Ø
    * NUPPER, MUPPER, W, IER)                                         IEG  2Ø
C THE INTEGRAL EQUATION BEING SOLVED IS                               IEG  3Ø
C                    B                                                IEG  4Ø
C        X(S) -  INT  KERNEL(S,T)*X(T)*DT = RHFCN(S)                  IEG  5Ø
C                    A                                                IEG  6Ø
C THE METHOD BEING USED IS BASED ON THE NYSTROM METHOD WITH           IEG  7Ø
C GAUSSIAN QUADRATURE, WITH AN ITERATIVE TECHNIQUE OF SOLUTION        IEG  8Ø
C FOR THE RESULTING LINEAR SYSTEM.                                    IEG  9Ø
C KERNEL     THESE ARE DOUBLE PRECISION FUNCTIONS OF TWO AND ONE      IEG  1ØØ
C RHFCN      VARIABLES, RESPECTIVELY. THEY MUST BE DECLARED IN AN     IEG  11Ø
C            EXTERNAL STATEMENT IN THE PROGRAM CALLING IEGAUS.        IEG  12Ø
C EP         THE DESIRED ERROR. THE VARIABLE EP IS CHANGED ON         IEG  13Ø
C            COMPLETION OF THE PROGRAM. SEE THE DISCUSSION OF IER     IEG  14Ø
```

```
C            AND IFLAG FOR MORE INFORMATION.                              IEG  15Ø
C IFLAG =Ø   EP IS INTERPRETED AS AN ABSOLUTE ERROR TOLERANCE.            IEG  16Ø
C       =1   EP IS INTERPRETED AS A RELATIVE ERROR TOLERANCE.             IEG  17Ø
C X          THE COMPUTED APPROXIMATE SOLUTION OF THE INTEGRAL            IEG  18Ø
C            EQUATION, EVALUATED AT THE NODE POINTS IN T, IS              IEG  19Ø
C            STORED IN X ON COMPLETION OF THE ROUTINE. THIS IS            IEG  2ØØ
C            TRUE IRREGARDLESS OF WHETHER OR NOT THE DESIRED ERROR        IEG  21Ø
C            TOLERANCE WAS ATTAINED.                                      IEG  22Ø
C T          CONTAINS THE NODE POINTS AT WHICH THE SOLUTION OF THE        IEG  23Ø
C            INTEGRAL EQUATION IS DESIRED. SEE THE VARIABLE NT FOR        IEG  24Ø
1C           MORE INFORMATION.                                            IEG  25Ø
C NT         IF NT=Ø, THEN T AND X WILL BE SET EQUAL TO THE FINAL         IEG  26Ø
C            GAUSSIAN NODES AND THE CORRESPONDING SOLUTION VALUES,        IEG  27Ø
C            AND NT WILL BE SET TO THE NUMBER OF THE SOLUTION             IEG  28Ø
C            VALUES STORED IN X AND T. THE ARRAYS T AND X SHOULD          IEG  29Ø
C            HAVE DIMENSION AT LEAST MUPPER, ASSIGNED IN THE              IEG  3ØØ
C            CALLING PROGRAM.                                             IEG  31Ø
C            IF NT .GT. Ø, THEN T CONTAINS NT USER SUPPLIED NODES         IEG  32Ø
C            AT WHICH THE SOLUTION X IS DESIRED.                          IEG  33Ø
C NUPPER     AN UPPER LIMIT ON THE VARIABLE N IN THIS PROGRAM.            IEG  34Ø
C            N IS THE ORDER OF A LINEAR SYSTEM WHICH IS BEING             IEG  35Ø
C            USED TO ITERATIVELY SOLVE A LARGER LINEAR SYSTEM OF          IEG  36Ø
C            ORDER M WHICH APPROXIMATES THE ABOVE INTEGRAL                IEG  37Ø
C            EQUATION.                                                    IEG  38Ø
C MUPPER     AN UPPER LIMIT ON THE VARIABLE M IN THE PROGRAM.             IEG  39Ø
C            N AND M ARE ALWAYS POWERS OF TWO.                            IEG  4ØØ
C W          TEMPORARY WORKING STORAGE FOR THE PROGRAM. IT MUST           IEG  41Ø
C            CONTAIN AT LEAST 5*NU*NU+9*(NU+MU) POSITIONS, WITH           IEG  42Ø
C            NU=NUPPER, MU=MUPPER.                                        IEG  43Ø
C IER =Ø     THIS ERROR COMPLETION CODE MEANS THE ROUTINE WAS             IEG  44Ø
C            COMPLETED SATISFACTORILY. EP CONTAINS THE PREDICTED          IEG  45Ø
C            ERROR.                                                       IEG  46Ø
C     =1     THE ERROR TEST WAS NOT SATISFIED. EP CONTAINS THE            IEG  47Ø
C            PREDICTED ERROR.                                             IEG  48Ø
C     =2     THE ERROR TEST WAS NOT SATISFIED. EP HAS BEEN SET            IEG  49Ø
C            TO ZERO.                                                     IEG  5ØØ
C     =3     THE ORIGINAL VALUE OF EP WAS TOO SMALL, DUE TO               IEG  51Ø
C            POSSIBLE ILL-CONDITIONING PROBLEMS IN THE INTEGRAL           IEG  52Ø
C            EQUATION. THE VALUE OF EP WAS RESET TO A MORE                IEG  53Ø
C            REALISTIC VALUE, AND THAT TOLERANCE WAS ATTAINED.            IEG  54Ø
C     =4     THE ERROR WAS SATISFACTORY AT THE GAUSSIAN NODE               IEG  55Ø
C            POINTS (IER=Ø), BUT THE INTERPOLATION PROCESS(DUE TO         IEG  56Ø
C            NT .GT. Ø) MAY NOT PRESERVE THIS ACCURACY. CHECK THE         IEG  57Ø
C            VALUE OF NORM(K) FOR POSSIBLE INDICATIONS THAT THE           IEG  58Ø
C            INTEGRAL EQUATION MAY BE ALMOST FIRST KIND. SUCH             IEG  59Ø
C            EQUATIONS ARE QUITE ILL-CONDITIONED. THE ERROR IN EP         IEG  6ØØ
C            IS THE PREDICTED ERROR FOR THE SOLUTION AT THE               IEG  61Ø
C            GAUSSIAN NODE POINTS OF ORDER MFINAL.                        IEG  62Ø
C     =5     THE ANALOGUE OF IER=4, BUT WITH IER=1 AT THE                 IEG  63Ø
C            GAUSSIAN NODE POINTS.                                        IEG  64Ø
C     =6     THE ANALOGUE OF IER=4, BUT WITH IER=3 AT THE                 IEG  65Ø
C            GAUSSIAN NODE POINTS.                                        IEG  66Ø
C IEGAUS IS PRESENTLY LIMITED TO NUPPER .LE. 1ØØ. THIS IS ENTIRELY        IEG  67Ø
C DUE TO A RESTRICTION IN THE PROGRAM LINSYS. TO REMOVE THIS              IEG  68Ø
C RESTRICTION, CHANGE THE DIMENSION STATEMENTS IN THE                     IEG  69Ø
C COMMON/LINEAR/ STATEMENTS IN LINSYS AND THE SUBPROGRAM IEGS.            IEG  7ØØ
      DOUBLE PRECISION KERNEL, RHFCN, A, B, EP, X, T, W, CUTOFF,          IEG  71Ø
     * ROOTRT, UNITRD, R1, R2, FINLEP, NORMK                             IEG  72Ø
      DIMENSION X(1), T(1), W(1)                                          IEG  73Ø
      EXTERNAL KERNEL, RHFCN                                              IEG  74Ø
C *********************************************************************  IEG  75Ø
C                                                                     *   IEG  76Ø
      COMMON /INFO/ R1, R2, FINLEP, NORMK, NFINAL, MFINAL                 IEG  77Ø
C                                                                     *   IEG  78Ø
C    THE NUMBERS IN INFO GIVE ADDITIONAL INFORMATION ABOUT THE        *   IEG  79Ø
C    FUNCTIONING OF IEGAUS. R1 IS THE ITERATIVE RATE OF CONVERGENCE   *   IEG  8ØØ
C    IN THE MOST RECENTLY COMPUTED LINEAR SYSTEM. R2 IS THE RATE OF   *   IEG  81Ø
C    CONVERGENCE OF THE GAUSSIAN QUADRATURE VARIANT OF THE NYSTROM    *   IEG  82Ø
C    METHOD. FINLEP IS THE FINAL VALUE OF EP USED AS THE DESIRED      *   IEG  83Ø
C    ERROR TOLERANCE. USUALLY FINLEP WILL EQUAL THE INPUT VALUE OF    *   IEG  84Ø
C    EP, UNLESS EP WAS MUCH TOO SMALL. NORMK IS AN APPROXIMATE        *   IEG  85Ø
C    VALUE FOR THE NORM OF THE INTEGRAL OPERATOR K, AND IT IS         *   IEG  86Ø
C    CALCULATED ONLY IF NT .GT. Ø.                                    *   IEG  87Ø
C    NFINAL AND MFINAL ARE THE FINAL VALUES OF N AND M USED IN        *   IEG  88Ø
C    IEGAUS. IF NFINAL=MFINAL, THEN ITERATION WAS NOT INVOKED         *   IEG  89Ø
```

```
C    SUCCESSFULLY.                                                    *   IEG  9ØØ
C                                                                     *   IEG  91Ø
C ********************************************************************   IEG  92Ø
C                                                                     *   IEG  93Ø
      DATA UNITRD /2.22D-16/                                              IEG  94Ø
C                                                                     *   IEG  95Ø
C    UNITRD IS THE SMALLEST NUMBER U FOR WHICH                        *   IEG  96Ø
C                  1+U .GT. 1 .                                       *   IEG  97Ø
C    UNITRD VARIES WITH THE COMPUTER AND WITH THE ARITHMETIC BEING    *   IEG  98Ø
C    USED. TO CHANGE TO ANOTHER ARITHMETIC, UNITRD MUST BE CHANGED.   *   IEG  99Ø
C    BUT UNITRD ALSO REFLECTS THE ACCURACY OF THE CONSTANTS USED      *   IEG 1ØØØ
C    IN SUBROUTINE WANDT.                                             *   IEG 1Ø1Ø
C                                                                     *   IEG 1Ø2Ø
C ********************************************************************   IEG 1Ø3Ø
      DATA CUTOFF /Ø.5DØ/, ROOTRT /Ø.1DØ/                                 IEG 1Ø4Ø
C SET UP THE RELATIVE BASE ADDRESSES FOR THE VARIOUS ARRAYS INTO          IEG 1Ø5Ø
C WHICH W IS TO BE SPLIT.                                                 IEG 1Ø6Ø
      N = NUPPER                                                          IEG 1Ø7Ø
      M = MUPPER                                                          IEG 1Ø8Ø
      NSQ = N*N                                                           IEG 1Ø9Ø
      I1 = 1                                                              IEG 11ØØ
      I2 = I1 + NSQ                                                       IEG 111Ø
      I3 = I2 + NSQ                                                       IEG 112Ø
      I4 = I3 + NSQ/2                                                     IEG 113Ø
      I5 = I4 + NSQ/2                                                     IEG 114Ø
      I6 = I5 + M                                                         IEG 115Ø
      I7 = I6 + M                                                         IEG 116Ø
      I8 = I7 + N                                                         IEG 117Ø
      I9 = I8 + N                                                         IEG 118Ø
      I1Ø = I9 + M                                                        IEG 119Ø
      I11 = I1Ø + N                                                       IEG 12ØØ
      I12 = I11 + M                                                       IEG 121Ø
      I13 = I12 + M                                                       IEG 122Ø
      I14 = I13 + M                                                       IEG 123Ø
      I15 = I14 + N                                                       IEG 124Ø
      I16 = I15 + M                                                       IEG 125Ø
      I17 = I16 + M                                                       IEG 126Ø
      I18 = I17 + N                                                       IEG 127Ø
      I19 = I18 + M                                                       IEG 128Ø
      I2Ø = I19 + 4*N                                                     IEG 129Ø
      I21 = I2Ø + NSQ                                                     IEG 13ØØ
      NHALF = N/2                                                         IEG 131Ø
      CALL IEGS(KERNEL, RHFCN, A, B, EP, IFLAG, X, T, NT, NUPPER,         IEG 132Ø
     * MUPPER, IER, CUTOFF, ROOTRT, UNITRD, NHALF, W(I1), W(I2),         IEG 133Ø
     * W(I3), W(I4), W(I5), W(I6), W(I7), W(I8), W(I9), W(I1Ø),           IEG 134Ø
     * W(I11), W(I12), W(I13), W(I14), W(I15), W(I16), W(I17),           IEG 135Ø
     * W(I18), W(I19), W(I2Ø), W(I21))                                   IEG 136Ø
      RETURN                                                              IEG 137Ø
      END                                                                 IEG 138Ø



      SUBROUTINE IEGS(KERNEL, RHFCN, A, B, EP, IFLAG, X, T, NT,           IEG   1Ø
     * NUP, MUP, IER, CUTOFF, ROOTRT, UNITRD, NHALF, LUFACT, KMM,         IEG   2Ø
     * KMN, KNM, RHS, R, RH, DELN, TM, TN, XM, XMZ, WM, WN, OLDX,         IEG   3Ø
     * SAVE, XN, SAVE2, ASIDE, ASIDE3, IMKNN)                             IEG   4Ø
C THIS ROUTINE CONTROLS THE SOLUTION OF THE INTEGRAL EQUATION.            IEG   5Ø
      DOUBLE PRECISION KERNEL, RHFCN, A, B, EP, X, T, CUTOFF,             IEG   6Ø
     * ROOTRT, UNITRD, LUFACT, KMM, KMN, KNM, RHS, R, RH, DELN, TM,       IEG   7Ø
     * TN, XM, XMZ, WM, WN, OLDX, SAVE, XN, SAVE2, ASIDE, ASIDE2,         IEG   8Ø
     * ASIDE3, IMKNN, RESID, SCALE, ELINSY, RELRSD, XNORM, PASTC,         IEG   9Ø
     * PASTRE, R1, R2, FINLEP, NORMK, DFLOAT, NORM, CONEW, RMIN,          IEG  1ØØ
     * R1RAT, COND, AVERR, EPS, DET, RELMIN, DMIN1, DMAX1, DSQRT,         IEG  11Ø
     * ERROR, NUMR2, DENR2, TEMP2, RELERR, NUMR1, DENR1, RATE,            IEG  12Ø
     * TEMP, RT, ESTERR, TEST                                             IEG  13Ø
      INTEGER FLAG, OLDM                                                  IEG  14Ø
      DIMENSION X(1), T(1), LUFACT(NUP,NUP), KMM(NUP,NUP),                IEG  15Ø
     * RHS(MUP), KNM(NHALF,NUP), KMN(NUP,NHALF), R(MUP), RH(NUP),         IEG  16Ø
     * DELN(NUP), TM(MUP), TN(NUP), XM(MUP), XMZ(MUP), WM(MUP),           IEG  17Ø
     * WN(NUP), OLDX(MUP), SAVE(MUP), XN(NUP), SAVE2(MUP),                IEG  18Ø
     * ASIDE(NUP,4), ASIDE2(5), ASIDE3(NUP,NUP), IMKNN(NUP,NUP)           IEG  19Ø
      COMMON /LINEAR/ RESID(1ØØ), SCALE(1ØØ), ELINSY, RELRSD,             IEG  2ØØ
     * IPIVOT(1ØØ)                                                        IEG  21Ø
      COMMON /INFO/ R1, R2, FINLEP, NORMK, NFINAL, MFINAL                 IEG  22Ø
      EXTERNAL KERNEL, RHFCN                                              IEG  23Ø
C INITIALIZATION                                                          IEG  24Ø
```

```
      LOOP = 1                                                          IEG  25Ø
      N = 2                                                             IEG  26Ø
      R2 = Ø.5DØ                                                        IEG  27Ø
      M = 2*N                                                           IEG  28Ø
      R1RAT = ROOTRT                                                    IEG  29Ø
      COND = 1.ØDØ                                                      IEG  3ØØ
      PASTC = 1.ØDØ                                                     IEG  31Ø
      PASTRE = Ø.ØDØ                                                    IEG  32Ø
      EPS = EP                                                          IEG  33Ø
C STAGE A. DIRECT SOLUTION OF LINEAR SYSTEM (I-KN)*XN=RHS, WHILE        IEG  34Ø
C TRYING TO FIND A GOOD APPROXIMATE INVERSE TO IMPLEMENT                IEG  35Ø
C ITERATIVE METHOD OF SOLUTION.                                         IEG  36Ø
C CREATE THE NODES AND WEIGHTS TN(I) AND WN(I), I=1,...,N               IEG  37Ø
      CALL WANDT(WN, TN, N, A, B)                                       IEG  38Ø
C SET UP MATRIX FOR (I-KN)*XN=RHFCN                                     IEG  39Ø
      DO 2Ø J=1,N                                                       IEG  4ØØ
        DO 1Ø I=1,N                                                     IEG  41Ø
          IMKNN(I,J) = -WN(J)*KERNEL(TN(I),TN(J))                       IEG  42Ø
   1Ø   CONTINUE                                                        IEG  43Ø
        XMZ(J) = RHFCN(TN(J))                                           IEG  44Ø
        IMKNN(J,J) = IMKNN(J,J) + 1.ØDØ                                 IEG  45Ø
   2Ø CONTINUE                                                          IEG  46Ø
      GO TO 6Ø                                                          IEG  47Ø
C THIS IS ENTRANCE FOR AN INCREASED VALUE OF N, USING PREVIOUSLY        IEG  48Ø
C STORED VALUES IN KMM TO DEFINE MATRIX FOR (I-KN)*XN=RHFCN WITH        IEG  49Ø
C NEW VALUE OF N.                                                       IEG  5ØØ
   3Ø DO 5Ø J=1,N                                                       IEG  51Ø
        DO 4Ø I=1,N                                                     IEG  52Ø
          IMKNN(I,J) = -KMM(I,J)                                        IEG  53Ø
   4Ø   CONTINUE                                                        IEG  54Ø
        WN(J) = WM(J)                                                   IEG  55Ø
        TN(J) = TM(J)                                                   IEG  56Ø
        XMZ(J) = RHS(J)                                                 IEG  57Ø
        IMKNN(J,J) = IMKNN(J,J) + 1.ØDØ                                 IEG  58Ø
   5Ø CONTINUE                                                          IEG  59Ø
C THIS IS THE ENTRANCE WHEN ITERATION IN STAGE B FAILS AND WE NEED      IEG  6ØØ
C TO INCREASE N TO OBTAIN A BETTER ITERATIVE RATE.                      IEG  61Ø
   6Ø CONTINUE                                                          IEG  62Ø
C    SOLVE (I-KN)*XN=RHFCN AT ALL TN(I).ALSO OBTAIN THE LU              IEG  63Ø
C    DECOMPOSITION FOR LATER USE IN THE STAGE B ITERATIVE METHOD.       IEG  64Ø
C ***************************************************************      IEG  65Ø
C                                                                  *    IEG  66Ø
      CALL LINSYS(IMKNN, LUFACT, N, XMZ, XN, 2, DET, NUP)               IEG  67Ø
C                                                                  *    IEG  68Ø
C    LINSYS IS A GENERAL LINEAR EQUATION SOLVER. IT HAS SPECIAL    *    IEG  69Ø
C    OPTIONS WHICH ARE USED IN THE FOLLOWING PROGRAM, AND THUS     *    IEG  7ØØ
C    LINSYS SHOULD NOT BE REPLACED BY ANOTHER LINEAR EQUATION      *    IEG  71Ø
C    PROGRAM. LINSYS IS ALSO USED IN THE SUBROUTINE ITERT.         *    IEG  72Ø
C                                                                  *    IEG  73Ø
C ***************************************************************      IEG  74Ø
      COND = CONEW(COND,ELINSY,RELRSD,AVERR,PASTC,PASTRE)               IEG  75Ø
      RELMIN = RMIN(N,N,COND,UNITRD,AVERR)                              IEG  76Ø
      IF (LOOP.EQ.1) GO TO 1ØØ                                          IEG  77Ø
      IF (LOOP.EQ.2) GO TO 8Ø                                           IEG  78Ø
C SET UP APPROXIMATE RATE OF CONVERGENCE OF SOLUTIONS XN TO TRUE        IEG  79Ø
C SOLUTION X. ALSO SET UP DESIRED RATIO FOR ITERATIVE METHOD.           IEG  8ØØ
      NUMR2 = NORM(XN,OLDX,N,1)                                         IEG  81Ø
      R2 = DMIN1(Ø.5DØ,DMAX1(NUMR2/DENR2,1.ØD-4))                       IEG  82Ø
      R1RAT = DMIN1(ROOTRT,DSQRT(R2))                                   IEG  83Ø
C CHECK FOR ERROR IN XN USING TEST INVOLVING R2 AND OLDX,ACCORDING      IEG  84Ø
C TO THEORY FOR ASYMPTOTIC ERROR BOUNDS. MODIFY ERROR IF IT IS          IEG  85Ø
C OUTSIDE PRECISION RANGE OF COMPUTER, POSSIBLY DUE TO                  IEG  86Ø
C ILL-CONDITIONING.                                                     IEG  87Ø
   7Ø ERROR = (R2/(1.ØDØ-R2))*NUMR2                                     IEG  88Ø
      XNORM = NORM(XN,XN,N,Ø)                                           IEG  89Ø
      RELERR = ERROR/XNORM                                              IEG  9ØØ
      IF (IFLAG.EQ.Ø) EPS = DMAX1(EP,XNORM*RELMIN)                      IEG  91Ø
      IF (IFLAG.EQ.1) EPS = DMAX1(EP,RELMIN)                            IEG  92Ø
      IF (IFLAG.EQ.1) ERROR = DMAX1(RELERR,RELMIN)                      IEG  93Ø
      IF ((IFLAG.EQ.Ø) .AND. (RELERR.LT.RELMIN)) ERROR =                IEG  94Ø
     * RELMIN*XNORM                                                     IEG  95Ø
      IF (ERROR.LE.EPS) GO TO 9Ø                                        IEG  96Ø
      DENR2 = NUMR2                                                     IEG  97Ø
      GO TO 1ØØ                                                         IEG  98Ø
C ENTRANCE FOR LOOP=2.                                                  IEG  99Ø
   8Ø NUMR2 = NORM(XN,OLDX,N,1)                                         IEG 1ØØØ
```

```
      DENR2 = Ø.ØDØ                                                    IEG 1Ø1Ø
      GO TO 7Ø                                                         IEG 1Ø2Ø
C EXIT FOR SUCCESSFUL RETURN. ITERATION WAS NOT NECESSARY.             IEG 1Ø3Ø
   9Ø CALL LEAVE(Ø, N, N, XN, TN, WN, ERROR, KERNEL, RHFCN, EP,        IEG 1Ø4Ø
     * IFLAG, X, T, NT, IER, EPS, ELINSY, TN, WN, TM, WM, XM, XMZ,     IEG 1Ø5Ø
     * KMM, KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN, NUP, NHALF,     IEG 1Ø6Ø
     * XNORM)                                                          IEG 1Ø7Ø
      RETURN                                                           IEG 1Ø8Ø
C ATTEMPT TO SOLVE (I-KM)*XM=RHFCN ITERATIVELY, CHECKING TO SEE IF     IEG 1Ø9Ø
C THE RATE OF CONVERGENCE IS SUFFICIENTLY FAST SO AS TO ENTER          IEG 11ØØ
C STAGE B.                                                             IEG 111Ø
C CALCULATE TM(I) AND WM(I), I=1,...,M.                                IEG 112Ø
  1ØØ CALL WANDT(WM, TM, M, A, B)                                      IEG 113Ø
      FLAG = Ø                                                         IEG 114Ø
C CALCULATE INITIAL GUESS XMZ FOR ITERATION METHOD.                    IEG 115Ø
      CALL INTERP(TM, WM, XMZ, M, TN, WN, XN, N, KERNEL, RHFCN,        IEG 116Ø
     * RHS, KMN, NHALF, NUP)                                           IEG 117Ø
      DO 11Ø I=1,M                                                     IEG 118Ø
        OLDX(I) = XMZ(I)                                               IEG 119Ø
  11Ø CONTINUE                                                         IEG 12ØØ
C CALCULATE FIRST ITERATE.                                             IEG 121Ø
      CALL ITERT(KERNEL, RHFCN, N, TN, WN, M, TM, WM, XM, XMZ, KMM,    IEG 122Ø
     * KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN, NUP, NHALF, FLAG)    IEG 123Ø
      COND = CONEW(COND,ELINSY,RELRSD,AVERR,PASTC,PASTRE)              IEG 124Ø
      DENR1 = NORM(XM,XMZ,M,1)                                         IEG 125Ø
      FLAG = 1                                                         IEG 126Ø
      DO 12Ø I=1,M                                                     IEG 127Ø
        XMZ(I) = XM(I)                                                 IEG 128Ø
  12Ø CONTINUE                                                         IEG 129Ø
C CALCULATE SECOND ITERATE.                                            IEG 13ØØ
      CALL ITERT(KERNEL, RHFCN, N, TN, WN, M, TM, WM, XM, XMZ, KMM,    IEG 131Ø
     * KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN, NUP, NHALF, FLAG)    IEG 132Ø
      COND = CONEW(COND,ELINSY,RELRSD,AVERR,PASTC,PASTRE)              IEG 133Ø
      NUMR1 = NORM(XM,XMZ,M,1)                                         IEG 134Ø
C CHECK ON THE SPEED OF CONVERGENCE OF ITERATIVE METHOD. IF IT IS       IEG 135Ø
C SUFFICIENTLY RAPID, THEN FIX N AND GO TO STAGE B.                    IEG 136Ø
      R1 = NUMR1/DENR1                                                 IEG 137Ø
      RATE = R1                                                        IEG 138Ø
      IF (M.GT.NUP) GO TO 17Ø                                          IEG 139Ø
      IF (R1.LE.R1RAT) GO TO 13Ø                                       IEG 14ØØ
C THE ITERATION DID NOT WORK WELL ENOUGH, AND STAGE A IS TO BE         IEG 141Ø
C REPEATED. RE-INITIALIZE FOR SOLVING (I-KN)*XN=RHFCN AGAIN            IEG 142Ø
C WITH A LARGER N.                                                     IEG 143Ø
      N = M                                                            IEG 144Ø
      LOOP = LOOP + 1                                                  IEG 145Ø
      M = 2*N                                                          IEG 146Ø
      GO TO 3Ø                                                         IEG 147Ø
C THE ITERATIVE RATE IS SUFFICIENTLY RAPID, AND CONTROL WILL GO TO     IEG 148Ø
C STAGE B. SAVE INFORMATION IN CASE STAGE B ABORTS AT A LARGER         IEG 149Ø
C VALUE OF M AND STAGE A HAS TO BE RETURNED TO.                        IEG 15ØØ
  13Ø DO 14Ø I=1,M                                                     IEG 151Ø
        ASIDE(I,1) = OLDX(I)                                           IEG 152Ø
        ASIDE(I,2) = WM(I)                                             IEG 153Ø
        ASIDE(I,3) = TM(I)                                             IEG 154Ø
        ASIDE(I,4) = RHS(I)                                            IEG 155Ø
  14Ø CONTINUE                                                         IEG 156Ø
      ASIDE2(1) = LOOP                                                 IEG 157Ø
      ASIDE2(3) = R2                                                   IEG 158Ø
      ASIDE2(4) = DENR2                                                IEG 159Ø
      ASIDE2(5) = R1RAT                                                IEG 16ØØ
      DO 16Ø J=1,M                                                     IEG 161Ø
        DO 15Ø I=1,M                                                   IEG 162Ø
          ASIDE3(I,J) = KMM(I,J)                                       IEG 163Ø
  15Ø   CONTINUE                                                       IEG 164Ø
  16Ø CONTINUE                                                         IEG 165Ø
C STAGE B. ITERATIVE METHOD OF SOLUTION OF (I-KM)*XM=RHS.              IEG 166Ø
  17Ø OLDM = N                                                         IEG 167Ø
      ASIDE2(2) = M                                                    IEG 168Ø
      IF (R1.LE.CUTOFF) GO TO 2ØØ                                      IEG 169Ø
C THE ITERATES ARE CONVERGING VERY SLOWLY OR NOT AT ALL. THUS          IEG 17ØØ
C RETURN WITHOUT FURTHER ATTEMPTS TO LESSEN THE ERROR.                 IEG 171Ø
      IF (LOOP.NE.1) GO TO 19Ø                                         IEG 172Ø
  18Ø CALL LEAVE(2, N, N, XN, TN, WN, Ø.ØDØ, KERNEL, RHFCN, EP,        IEG 173Ø
     * IFLAG, X, T, NT, IER, EPS, ELINSY, TN, WN, TM, WM, XM, XMZ,     IEG 174Ø
     * KMM, KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN, NUP, NHALF,     IEG 175Ø
```

```
     * XNORM)                                                          IEG 176Ø
      RETURN                                                          IEG 177Ø
  19Ø CALL LEAVE(1, N, N, XN, TN, WN, ERROR, KERNEL, RHFCN, EP,       IEG 178Ø
     * IFLAG, X, T, NT, IER, EPS, ELINSY, TN, WN, TM, WM, XM, XMZ,    IEG 179Ø
     * KMM, KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN, NUP, NHALF,   IEG 18ØØ
     * XNORM)                                                         IEG 181Ø
      RETURN                                                          IEG 182Ø
C TEST TO SEE IF THE CURRENT ITERATE XM IS SUFFICIENTLY ACCURATE      IEG 183Ø
C COMPARED TO THE TRUE XM.                                            IEG 184Ø
  2ØØ RATE = R1*RATE                                                  IEG 185Ø
      TEMP = NORM(XM,OLDX,M,1)                                        IEG 186Ø
      IF (LOOP.EQ.1) TEMP2 = Ø.5DØ                                    IEG 187Ø
      IF (LOOP.GT.1) TEMP2 = TEMP/DENR2                               IEG 188Ø
      RT = DMIN1(Ø.Ø1DØ,DMAX1(TEMP2,Ø.ØØØ1DØ))/2.ØDØ                  IEG 189Ø
      XNORM = NORM(XM,XM,M,Ø)                                         IEG 19ØØ
      ESTERR = (RT/(1.ØDØ-RT))*TEMP/XNORM                             IEG 191Ø
      IF (ESTERR.LT.RELMIN) ESTERR = RELMIN                           IEG 192Ø
      ESTERR = ESTERR*XNORM                                           IEG 193Ø
      TEST = ((1.ØDØ-R1)/R1)*ESTERR                                   IEG 194Ø
      IF (NUMR1.LE.TEST) GO TO 26Ø                                    IEG 195Ø
C ITERATE NOT SUFFICIENTLY ACCURATE. INITIALIZE FOR COMPUTATION       IEG 196Ø
C OF ANOTHER ITERATE.                                                 IEG 197Ø
      DENR1 = NUMR1                                                   IEG 198Ø
      DO 21Ø I=1,M                                                    IEG 199Ø
        XMZ(I) = XM(I)                                                IEG 2ØØØ
  21Ø CONTINUE                                                        IEG 2Ø1Ø
      CALL ITERT(KERNEL, RHFCN, N, TN, WN, M, TM, WM, XM, XMZ, KMM,   IEG 2Ø2Ø
     * KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN, NUP, NHALF, FLAG)   IEG 2Ø3Ø
      COND = CONEW(COND,ELINSY,RELRSD,AVERR,PASTC,PASTRE)             IEG 2Ø4Ø
      NUMR1 = NORM(XM,XMZ,M,1)                                        IEG 2Ø5Ø
      R1 = NUMR1/DENR1                                                IEG 2Ø6Ø
      IF (R1.LE.CUTOFF) GO TO 2ØØ                                     IEG 2Ø7Ø
C THIS IS ENTRANCE FOR CASE WHERE ITERATION FAILS IN STAGE B.          IEG 2Ø8Ø
C PARAMETERS MUST BE RESET FOR A RETURN TO STAGE A OR FOR AN          IEG 2Ø9Ø
C ABORTIVE EXIT IF N CANNOT BE INCREASED ANY FURTHER.                 IEG 21ØØ
  22Ø MNEW = ASIDE2(2)                                                IEG 211Ø
      IF (MNEW.GT.NUP) GO TO 25Ø                                      IEG 212Ø
      N = MNEW                                                        IEG 213Ø
      DO 24Ø J=1,N                                                    IEG 214Ø
        DO 23Ø I=1,N                                                  IEG 215Ø
          IMKNN(I,J) = -ASIDE3(I,J)                                   IEG 216Ø
  23Ø   CONTINUE                                                      IEG 217Ø
        OLDX(J) = ASIDE(J,1)                                          IEG 218Ø
        WN(J) = ASIDE(J,2)                                            IEG 219Ø
        TN(J) = ASIDE(J,3)                                            IEG 22ØØ
        XMZ(J) = ASIDE(J,4)                                           IEG 221Ø
        IMKNN(J,J) = IMKNN(J,J) + 1.ØDØ                               IEG 222Ø
  24Ø CONTINUE                                                        IEG 223Ø
      M = 2*N                                                         IEG 224Ø
      LOOP = ASIDE2(1) + 1.ØDØ                                        IEG 225Ø
      R2 = ASIDE2(3)                                                  IEG 226Ø
      DENR2 = ASIDE2(4)                                               IEG 227Ø
      R1RAT = ASIDE2(5)                                               IEG 228Ø
      GO TO 6Ø                                                        IEG 229Ø
C ABORTIVE EXIT FROM STAGE B. N CANNOT BE INCREASED FURTHER, AND      IEG 23ØØ
C R1 IS NOT SUFFICIENTLY SMALL.                                       IEG 231Ø
  25Ø IF (LOOP.EQ.1) GO TO 18Ø                                        IEG 232Ø
      CALL WANDT(WM, TM, OLDM, A, B)                                  IEG 233Ø
      CALL LEAVE(1, N, OLDM, SAVE, TM, WM, ERROR, KERNEL, RHFCN,      IEG 234Ø
     * EP, IFLAG, X, T, NT, IER, EPS, ELINSY, TN, WN, TM, WM, XM,     IEG 235Ø
     * XMZ, KMM, KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN, NUP,     IEG 236Ø
     * NHALF, XNORM)                                                  IEG 237Ø
      RETURN                                                          IEG 238Ø
C AN ACCURATE VALUE OF XM HAS BEEN OBTAINED. R2 IS TO BE TESTED AS    IEG 239Ø
C TO WHETHER IT SHOULD BE RESET. THEN CHECK ERROR IN XM COMPARED      IEG 24ØØ
C WITH THE TRUE SOLUTION X.                                           IEG 241Ø
  26Ø IF (LOOP.EQ.1) GO TO 29Ø                                        IEG 242Ø
      NUMR2 = TEMP                                                    IEG 243Ø
      R2 = DMAX1(1.ØD-4,RATE,DMIN1(NUMR2/DENR2,Ø.5DØ))                IEG 244Ø
      DENR2 = NUMR2                                                   IEG 245Ø
  27Ø ERROR = (R2/(1.ØDØ-R2))*TEMP                                    IEG 246Ø
      XNORM = NORM(XM,XM,M,Ø)                                         IEG 247Ø
      RELERR = ERROR/XNORM                                            IEG 248Ø
      RELMIN = RMIN(N,M,COND,UNITRD,AVERR)                            IEG 249Ø
      IF (IFLAG.EQ.Ø) EPS = DMAX1(EP,XNORM*RELMIN)                    IEG 25ØØ
      IF (IFLAG.EQ.1) EPS = DMAX1(EP,RELMIN)                          IEG 251Ø
```

```
      IF (IFLAG.EQ.1) ERROR = DMAX1(RELERR,RELMIN)                    IEG 252Ø
      IF ((IFLAG.EQ.Ø) .AND. (RELERR.LT.RELMIN)) ERROR =              IEG 253Ø
     * RELMIN*XNORM                                                   IEG 254Ø
      IF (ERROR.GT.EPS) GO TO 28Ø                                     IEG 255Ø
      CALL LEAVE(Ø, N, M, XM, TM, WM, ERROR, KERNEL, RHFCN, EP,       IEG 256Ø
     * IFLAG, X, T, NT, IER, EPS, ELINSY, TN, WN, TM, WM, XM, XMZ,    IEG 257Ø
     * KMM, KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN, NUP, NHALF,    IEG 258Ø
     * XNORM)                                                         IEG 259Ø
      RETURN                                                          IEG 26ØØ
  28Ø MNEW = 2*M                                                      IEG 261Ø
      IF (MNEW.LE.MUP) GO TO 3ØØ                                      IEG 262Ø
C M CANNOT BE INCREASED ANY FURTHER.                                  IEG 263Ø
      CALL LEAVE(1, N, M, XM, TM, WM, ERROR, KERNEL, RHFCN, EP,       IEG 264Ø
     * IFLAG, X, T, NT, IER, EPS, ELINSY, TN, WN, TM, WM, XM, XMZ,    IEG 265Ø
     * KMM, KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN, NUP, NHALF,    IEG 266Ø
     * XNORM)                                                         IEG 267Ø
      RETURN                                                          IEG 268Ø
  29Ø DENR2 = TEMP                                                    IEG 269Ø
      LOOP = 2                                                        IEG 27ØØ
      GO TO 27Ø                                                       IEG 271Ø
C ERROR NOT SUFFICIENTLY SMALL. M IS INCREASED AND TWO MORE           IEG 272Ø
C MORE ITERATES ARE COMPUTED WITH THE NEW M.                          IEG 273Ø
  3ØØ OLDM = M                                                        IEG 274Ø
      M = MNEW                                                        IEG 275Ø
      DO 31Ø I=1,OLDM                                                 IEG 276Ø
        SAVE2(I) = WM(I)                                              IEG 277Ø
        SAVE(I) = TM(I)                                               IEG 278Ø
  31Ø CONTINUE                                                        IEG 279Ø
      CALL WANDT(WM, TM, M, A, B)                                     IEG 28ØØ
      FLAG = Ø                                                        IEG 281Ø
      CALL INTERP(TM, WM, XMZ, M, SAVE, SAVE2, XM, OLDM, KERNEL,      IEG 282Ø
     * RHFCN, RHS, KMN, NHALF, NUP)                                   IEG 283Ø
      DO 32Ø I=1,OLDM                                                 IEG 284Ø
        SAVE(I) = XM(I)                                               IEG 285Ø
  32Ø CONTINUE                                                        IEG 286Ø
      DO 33Ø I=1,M                                                    IEG 287Ø
        OLDX(I) = XMZ(I)                                              IEG 288Ø
  33Ø CONTINUE                                                        IEG 289Ø
      CALL ITERT(KERNEL, RHFCN, N, TN, WN, M, TM, WM, XM, XMZ, KMM,   IEG 29ØØ
     * KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN, NUP, NHALF, FLAG)   IEG 291Ø
      COND = CONEW(COND,ELINSY,RELRSD,AVERR,PASTC,PASTRE)             IEG 292Ø
      DENR1 = NORM(XM,XMZ,M,1)                                        IEG 293Ø
      FLAG = 1                                                        IEG 294Ø
      DO 34Ø I=1,M                                                    IEG 295Ø
        XMZ(I) = XM(I)                                                IEG 296Ø
  34Ø CONTINUE                                                        IEG 297Ø
      CALL ITERT(KERNEL, RHFCN, N, TN, WN, M, TM, WM, XM, XMZ, KMM,   IEG 298Ø
     * KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN, NUP, NHALF, FLAG)   IEG 299Ø
      COND = CONEW(COND,ELINSY,RELRSD,AVERR,PASTC,PASTRE)             IEG 3ØØØ
      NUMR1 = NORM(XM,XMZ,M,1)                                        IEG 3Ø1Ø
      R1 = NUMR1/DENR1                                                IEG 3Ø2Ø
      RATE = R1                                                       IEG 3Ø3Ø
      IF (R1.LE.CUTOFF) GO TO 2ØØ                                     IEG 3Ø4Ø
      GO TO 22Ø                                                       IEG 3Ø5Ø
      END                                                             IEG 3Ø6Ø



      SUBROUTINE WANDT(WV, TV, N, A, B)                               WAN   1Ø
C INTEGRATION WEIGHTS AND NODES ARE TO BE CALCULATED AND STORED IN    WAN   2Ø
C WV AND TV, RESPECTIVELY. N IS ASSUMED TO BE A POWER OF TWO. IF      WAN   3Ø
C 2 .LE. N .LE. 256, THEN GAUSSIAN QUADRATURE IS USED. IF N .GT.      WAN   4Ø
C 256, THEN THE INTERVAL (A,B) IS DIVIDED N/256 TIMES AND THE 256     WAN   5Ø
C POINT FORMULA IS APPLIED TO EACH SUBINTERVAL.                       WAN   6Ø
      DOUBLE PRECISION WV, TV, A, B, W, T, FLOOP, H, SCALE, AL, BL,   WAN   7Ø
     * S, R, FL                                                       WAN   8Ø
      DIMENSION WV(N), TV(N)                                          WAN   9Ø
      DIMENSION W(255), T(255)                                        WAN  1ØØ
      DATA T(1), T(2), T(3), T(4), T(5), T(6), T(7), T(8), T(9),      WAN  11Ø
     * T(1Ø), T(11), T(12), T(13), T(14), T(15)                       WAN  12Ø
     * /.5773502691896260,.8611363115940530,.3399810435848560,       WAN  13Ø
     * .9602898564975360,.7966664774136270,.5255324099163290,        WAN  14Ø
     * .1834346424956500,.9894009349916500,.9445750230732330,        WAN  15Ø
     * .8656312023878320,.7554044083550030,.6178762444026440,        WAN  16Ø
     * .4580167776572270,.2816035507792590,.9501250983763740-1/      WAN  17Ø
      DATA T(16), T(17), T(18), T(19), T(2Ø), T(21), T(22), T(23),    WAN  18Ø
```

```
* T(24), T(25), T(26), T(27), T(28), T(29), T(3Ø), T(31)          WAN  19Ø
* /.997263861849482DØ,.985611511545268DØ,.964762255587506DØ,     WAN  2ØØ
* .9349Ø6Ø7593774ØDØ,.896321155766Ø52DØ,.849367613732570DØ,      WAN  21Ø
* .794483795967942DØ,.732182118740290DØ,.6630442669302 15DØ,     WAN  22Ø
* .58771575724Ø762DØ,.5Ø68999Ø8932229DØ,.42135127613Ø635DØ,      WAN  23Ø
* .3318686Ø2282128DØ,.239287362252137DØ,.144471961582796DØ,      WAN  24Ø
* .483Ø76656877383D-1/                                            WAN  25Ø
 DATA T(32), T(33), T(34), T(35), T(36), T(37), T(38), T(39),     WAN  26Ø
* T(4Ø), T(41), T(42), T(43), T(44), T(45), T(46), T(47)          WAN  27Ø
* /.9993Ø5Ø41735772DØ,.99634Ø116771955DØ,.991Ø13371476744DØ,     WAN  28Ø
* .983336253884626DØ,.973326827789911DØ,.961ØØ8799652Ø54DØ,      WAN  29Ø
* .946411374858403DØ,.929569172131940DØ,.91Ø522137Ø785Ø3DØ,      WAN  3ØØ
* .889315445995114DØ,.865999398154Ø93DØ,.84Ø629296252580DØ,      WAN  31Ø
* .813265315122798DØ,.783972358943341DØ,.7528199Ø726Ø532DØ,      WAN  32Ø
* .7198818501716 11DØ/                                           WAN  33Ø
 DATA T(48), T(49), T(5Ø), T(51), T(52), T(53), T(54), T(55),     WAN  34Ø
* T(56), T(57), T(58), T(59), T(6Ø), T(61), T(62), T(63)          WAN  35Ø
* /.685236313Ø54233DØ,.648965471254657DØ,.611155355172393DØ,     WAN  36Ø
* .571895646Ø2634DØ,.53127946401989 5DØ,.4894Ø31457Ø7Ø53DØ,      WAN  37Ø
* .446366Ø17253464DØ,.4Ø227Ø157963992DØ,.35722Ø158337668DØ,      WAN  38Ø
* .311322871990211DØ,.26468716220 8767DØ,.2174236437400Ø7DØ,     WAN  39Ø
* .1696444Ø423993DØ,.121462819296121DØ,.729931217877990D-1,      WAN  4ØØ
* .2435Ø2926634244D-1/                                           WAN  41Ø
 DATA T(64), T(65), T(66), T(67), T(68), T(69), T(7Ø), T(71),     WAN  42Ø
* T(72), T(73), T(74), T(75), T(76), T(77), T(78), T(79)          WAN  43Ø
* /.999824887947132DØ,.9990774599773 76DØ,.997733248625514DØ,    WAN  44Ø
* .995792758534981DØ,.99325711290Ø213DØ,.990127818491734DØ,      WAN  45Ø
* .9864Ø6742724586DØ,.982Ø961Ø8435719DØ,.977198491463907DØ,      WAN  46Ø
* .971716818747137DØ,.965654366431965DØ,.959Ø147578537ØØDØ,     WAN  47Ø
* .9518Ø1961341264DØ,.944Ø2Ø28783Ø22ØDØ,.935674388277916DØ,     WAN  48Ø
* .9267692 5Ø878948DØ/                                           WAN  49Ø
 DATA T(8Ø), T(81), T(82), T(83), T(84), T(85), T(86), T(87),    WAN  5ØØ
* T(88), T(89), T(9Ø), T(91), T(92), T(93), T(94), T(95),        WAN  51Ø
* T(96), T(97) /.91731Ø198Ø8Ø961DØ,.9Ø73Ø2883Ø1757DØ,           WAN  52Ø
* .896753288Ø49158DØ,.885667717345397DØ,.874Ø5279695 8Ø32DØ,    WAN  53Ø
* .861915468939548DØ,.849262987577969DØ,.8361Ø2915Ø6Ø9Ø7DØ,     WAN  54Ø
* .822443116955644DØ,.8Ø829175707914DØ,.793657294762193DØ,      WAN  55Ø
* .778548475Ø6412DØ,.76297433ØØ44Ø95DØ,.746944166797Ø62DØ,      WAN  56Ø
* .73Ø4675667419Ø9DØ,.713554377683587DØ,.6962147Ø8369514DØ,     WAN  57Ø
* .678458922447719DØ/                                            WAN  58Ø
 DATA T(98), T(99), T(1ØØ), T(1Ø1), T(1Ø2), T(1Ø3), T(1Ø4),      WAN  59Ø
* T(1Ø5), T(1Ø6), T(1Ø7), T(1Ø8), T(1Ø9), T(11Ø), T(111),         WAN  6ØØ
* T(112), T(113), T(114) /.66Ø297632272646DØ,                    WAN  61Ø
* .64174169256 23Ø8DØ,.6228Ø219391Ø585DØ,.6Ø349Ø456158549DØ,     WAN  62Ø
* .583818Ø21628763DØ,.563796648226618DØ,.5434383Ø241281ØDØ,     WAN  63Ø
* .5227551520511 75DØ,.5Ø1759559136144DØ,.48Ø464Ø724Ø4172DØ,    WAN  64Ø
* .458881419833552DØ,.437Ø245Ø1Ø37104DØ,.4149Ø6379552275DØ,     WAN  65Ø
* .39254Ø275Ø33267DØ,.369939555349859DØ,.347117728597636DØ,     WAN  66Ø
* .324Ø88435Ø24413DØ/                                            WAN  67Ø
 DATA T(115), T(116), T(117), T(118), T(119), T(12Ø), T(121),    WAN  68Ø
* T(122), T(123), T(124), T(125), T(126), T(127)                 WAN  69Ø
* /.3ØØ865438877677DØ,.27746262Ø1779Ø4DØ,.253893966422694DØ,    WAN  7ØØ
* .23Ø17356422666ØDØ,.2Ø631559Ø9Ø2Ø79DØ,.182334Ø5985337DØ,      WAN  71Ø
* .158244Ø42714225DØ,.134Ø59199461188DØ,.1Ø9794231127644DØ,     WAN  72Ø
* .8546364Ø5Ø45155D-1,.61Ø819696Ø41396D-1,.3666379Ø9687335D-1,  WAN  73Ø
* .122236989Ø6158D-1/                                            WAN  74Ø
 DATA T(128), T(129), T(13Ø), T(131), T(132), T(133), T(134),    WAN  75Ø
* T(135), T(136), T(137), T(138), T(139), T(14Ø), T(141),        WAN  76Ø
* T(142) /.999956Ø5ØØ18992DØ,.99976843740 9263DØ,               WAN  77Ø
* .99943Ø937466261DØ,.9989435258434Ø9DØ,.9983Ø6266473ØØ6DØ,     WAN  78Ø
* .997519252756721DØ,.9965826Ø2Ø23382DØ,.995496454481Ø96DØ,     WAN  79Ø
* .9942Ø97292241ØDØ,.9928763426Ø8822DØ,.991342771207583DØ,      WAN  8ØØ
* .98966Ø488745Ø65DØ,.987829747564861DØ,.98585Ø822286126DØ,     WAN  81Ø
* .983724ØØ976Ø315DØ/                                            WAN  82Ø
 DATA T(143), T(144), T(145), T(146), T(147), T(148), T(149),    WAN  83Ø
* T(15Ø), T(151), T(152), T(153), T(154), T(155), T(156),        WAN  84Ø
* T(157) /.9814496290 25464DØ,.979Ø28Ø21257622DØ,               WAN  85Ø
* .976459549719234DØ,.9737445997Ø437ØDØ,.97Ø883578480743DØ,     WAN  86Ø
* .967876915228489DØ,.964725Ø6Ø9757Ø6DØ,.961428488530732DØ,     WAN  87Ø
* .957987692411178DØ,.9544Ø3188769716DØ,.95Ø675515316628DØ,     WAN  88Ø
* .9468Ø5231239127DØ,.942792917117462DØ,.938639174837814DØ,     WAN  89Ø
* .9343446275Ø2ØØ3DØ/                                           WAN  9ØØ
 DATA T(158), T(159), T(16Ø), T(161), T(162), T(163), T(164),    WAN  91Ø
* T(165), T(166), T(167), T(168), T(169), T(17Ø), T(171),        WAN  92Ø
* T(172) /.9299Ø9919334Ø06DØ,.9253357155833 16DØ,               WAN  93Ø
* .92Ø6227Ø2425146DØ,.915771586857490DØ,.91Ø783Ø96595Ø65DØ,     WAN  94Ø
```

```
* .9Ø565797996Ø145DØ,.9ØØ397ØØ577Ø3Ø4DØ,.895ØØØ963223Ø85DØ,        WAN  95Ø
* .88947Ø661777611DØ,.8838Ø6931Ø33158DØ,.878Ø1Ø62Ø6Ø47Ø7DØ,        WAN  96Ø
* .872Ø82599995488DØ,.866Ø23758466555DØ,.859835ØØ49Ø3376DØ,        WAN  97Ø
* .853517267679503DØ/                                               WAN  98Ø
 DATA T(173), T(174), T(175), T(176), T(177), T(178), T(179),       WAN  99Ø
* T(18Ø), T(181), T(182), T(183), T(184), T(185), T(186),           WAN 1ØØ
* T(187) /.847Ø71494517296DØ,.84Ø498652345763DØ,                    WAN 1Ø1Ø
* .833799727155505DØ,.826975723850813DØ,.82ØØ27666Ø98917DØ,        WAN 1Ø2Ø
* .812956596176432DØ,.805763574812999DØ,.798449681Ø32171DØ,        WAN 1Ø3Ø
* .791Ø16Ø11989546DØ,.783463682808184DØ,.775793826411326DØ,        WAN 1Ø4Ø
* .768ØØ7593352446DØ,.76Ø1Ø6151642655DØ,.752Ø9Ø686575492DØ,        WAN 1Ø5Ø
* .7439624ØØ549112DØ/                                               WAN 1Ø6Ø
 DATA T(188), T(189), T(19Ø), T(191), T(192), T(193), T(194),       WAN 1Ø7Ø
* T(195), T(196), T(197), T(198), T(199), T(2ØØ), T(2Ø1),           WAN 1Ø8Ø
* T(2Ø2) /.735722512885918DØ,.727372259649652DØ,                    WAN 1Ø9Ø
* .718912893459971DØ,.71Ø345683Ø4543DØ,.7Ø1671914348685DØ,         WAN 11ØØ
* .692892887742577DØ,.684ØØ992Ø426Ø76DØ,.675Ø24344931163DØ,        WAN 111Ø
* .6659375Ø9182Ø49DØ,.65675Ø776292973DØ,.647465524363725DØ,        WAN 112Ø
* .638Ø83146272911DØ,.6286Ø5Ø49469Ø15DØ,.619Ø32655759261DØ,        WAN 113Ø
* .6Ø9367Ø1Ø96334DØ/                                                WAN 114Ø
 DATA T(2Ø3), T(2Ø4), T(2Ø5), T(2Ø6), T(2Ø7), T(2Ø8), T(2Ø9),       WAN 115Ø
* T(21Ø), T(211), T(212), T(213), T(214), T(215), T(216),           WAN 116Ø
* T(217) /.599610735362968DØ,.589764122154454DØ,                    WAN 117Ø
* .579829Ø38559Ø83DØ,.5698Ø6974936569DØ,.559699434694481DØ,        WAN 118Ø
* .5495Ø7934Ø62719DØ,.539234ØØ1866Ø59DØ,.528879179294822DØ,        WAN 119Ø
* .518445Ø19673674DØ,.5Ø7933Ø88228616DØ,.497344961852181DØ,        WAN 12ØØ
* .48668222886689ØDØ,.475946488786983DØ,.465139352Ø78479DØ,       WAN 121Ø
* .454262439917590DØ/                                               WAN 122Ø
 DATA T(218), T(219), T(22Ø), T(221), T(222), T(223), T(224),       WAN 123Ø
* T(225), T(226), T(227), T(228), T(229), T(23Ø), T(231),           WAN 124Ø
* T(232) /.443317383947527DØ,.4323Ø5826Ø33741DØ,                    WAN 125Ø
* .421229418Ø17624DØ,.41ØØ89821468717DØ,.3988887Ø7435459DØ,        WAN 126Ø
* .387627756194516DØ,.3763Ø8656998716DØ,.3649331Ø7823654DØ,        WAN 127Ø
* .3535Ø281511297ØDØ,.342Ø19493522372DØ,.33Ø484865662417DØ,        WAN 128Ø
* .3189ØØ66184Ø1Ø6DØ,.3Ø7268619799319DØ,.29559Ø48446Ø136DØ,       WAN 129Ø
* .283868Ø07657Ø82DØ/                                               WAN 13ØØ
 DATA T(233), T(234), T(235), T(236), T(237), T(238), T(239),       WAN 131Ø
* T(24Ø), T(241), T(242), T(243), T(244), T(245), T(246),           WAN 132Ø
* T(247) /.2721Ø2947876337DØ,.26Ø297Ø69991943DØ,                    WAN 133Ø
* .248452145ØØ1Ø57DØ,.236569949758284DØ,.22465226670Ø9132DØ,       WAN 134Ø
* .2127ØØ883622626DØ,.2ØØ717593323127DØ,.1887Ø4193421389DØ,        WAN 135Ø
* .176662486Ø449Ø2DØ,.164594277567554DØ,.1525Ø1378338656DØ,        WAN 136Ø
* .14Ø3856Ø2411376DØ,.1282487672706Ø7DØ,.116Ø92693560333DØ,        WAN 137Ø
* .1Ø3919Ø2481Ø5Ø9DØ/                                               WAN 138Ø
 DATA T(248), T(249), T(25Ø), T(251), T(252), T(253), T(254),       WAN 139Ø
* T(255) /.9173Ø1271635196D-1,.795272891ØØ233ØD-1,                  WAN 14ØØ
* .673125211657164D-1,.55Ø87655694634ØD-1,.428545265363791D-1,      WAN 141Ø
* .3Ø6149687799790D-1,.1837Ø8184788137D-1,.612391237518953D-2/      WAN 142Ø
 DATA W(1), W(2), W(3), W(4), W(5), W(6), W(7), W(8), W(9),         WAN 143Ø
* W(1Ø), W(11), W(12), W(13), W(14), W(15)                          WAN 144Ø
* /1.ØDØ,.347854845137454DØ,.652145154862546DØ,                    WAN 145Ø
* .1Ø1228536290376DØ,.222381Ø34453374DØ,.3137Ø6645877887DØ,        WAN 146Ø
* .362683783378362DØ,.2715245941175 41D-1,.622535239386479D-1,     WAN 147Ø
* .951585116824928D-1,.124628971255534DØ,.149595988816577DØ,       WAN 148Ø
* .1691565193950Ø3DØ,.1826Ø3415Ø44924DØ,.18945Ø61Ø455Ø68DØ/        WAN 149Ø
 DATA W(16), W(17), W(18), W(19), W(2Ø), W(21), W(22), W(23),       WAN 15ØØ
* W(24), W(25), W(26), W(27), W(28), W(29), W(3Ø), W(31)           WAN 151Ø
* /.7Ø1861ØØØ947Ø1ØD-2,.16274394730Ø957D-1,.25392Ø653Ø92621D-1,    WAN 152Ø
* .342738629130214D-1,.428358980222267D-1,.5Ø998Ø592623762D-1,     WAN 153Ø
* .586840934785355D-1,.658222227763618D-1,.723457941Ø88485D-1,     WAN 154Ø
* .781938957870703D-1,.833119242269468D-1,.87652Ø93ØØ44Ø38D-1,     WAN 155Ø
* .911738786957639D-1,.938443990808Ø46D-1,.9563872ØØ792749D-1,     WAN 156Ø
* .96540Ø885147278D-1/                                              WAN 157Ø
 DATA W(32), W(33), W(34), W(35), W(36), W(37), W(38), W(39),       WAN 158Ø
* W(4Ø), W(41), W(42), W(43), W(44), W(45), W(46), W(47)           WAN 159Ø
* /.178328Ø72169643D-2,.4147Ø3326Ø56247D-2,.65Ø445796897836D-2,    WAN 16ØØ
* .884675982636395D-2,.1116813946Ø1311D-1,.13463Ø478967186D-1,     WAN 161Ø
* .15726Ø3Ø476Ø247D-1,.179517157756973D-1,.2Ø134823153530 2D-1,    WAN 162Ø
* .2227Ø1738Ø83833D-1,.243527Ø25687109D-1,.263774697150547D-1,     WAN 163Ø
* .283396726142595D-1,.3Ø2346570724Ø25D-1,.32Ø579283548516D-1,     WAN 164Ø
* .338Ø51618371416D-1/                                              WAN 165Ø
 DATA W(48), W(49), W(5Ø), W(51), W(52), W(53), W(54), W(55),       WAN 166Ø
* W(56), W(57), W(58), W(59), W(6Ø), W(61), W(62), W(63)           WAN 167Ø
* /.354722132568824D-1,.37Ø5512854Ø24ØØD-1,.3855Ø1531786156D-1,    WAN 168Ø
* .39953741132723D-1,.412625632426235D-1,.424735151236536D-1,      WAN 169Ø
* .435837245293235D-1,.4459Ø5581637566D-1,.454916279274181D-1,     WAN 17ØØ
```

```
* .4628479658131440-1,.4696818281621ØØD-1,.4754Ø165714830 3D-1,     WAN 171Ø
* .479993885964583D-1,.48344762234803ØD-1,.485754674415Ø34D-1,      WAN 172Ø
* .4869Ø957ØØ91397D-1/                                              WAN 173Ø
 DATA W(64), W(65), W(66), W(67), W(68), W(69), W(7Ø), W(71),       WAN 174Ø
* W(72), W(73), W(74), W(75), W(76), W(77), W(78), W(79)            WAN 175Ø
* /.44938Ø96Ø292Ø9ØD-3,.1Ø458126793 4Ø35D-2,.16425Ø3Ø18669Ø3D-2,   WAN 176Ø
* .223828843096262D-2,.283275147145799D-2,.3425526Ø4Ø91Ø22D-2,     WAN 177Ø
* .4Ø1625498373864D-2,.46Ø45842567Ø296D-2,.519Ø16183267633D-2,     WAN 178Ø
* .577263754286570D-2,.635166316170719D-2,.692689256689881D-2,     WAN 179Ø
* .749798192563473D-2,.8Ø6458989Ø486Ø6D-2,.862637779861675D-2,     WAN 18ØØ
* .9183ØØ987166Ø87D-2/                                             WAN 181Ø
 DATA W(8Ø), W(81), W(82), W(83), W(84), W(85), W(86), W(87),       WAN 182Ø
* W(88), W(89), W(9Ø), W(91), W(92), W(93), W(94), W(95)            WAN 183Ø
* /.97341534150Ø681D-2,.1Ø279479Ø158322D-1,.1Ø81866Ø7395Ø31D-1,    WAN 184Ø
* .1135137632408Ø4D-1,.118773Ø73727403D-1,.1239613954395Ø9D-1,     WAN 185Ø
* .129Ø75627392673D-1,.134112712886163D-1,.139Ø69641329520D-1,     WAN 186Ø
* .1439434500041668D-1,.148731226021473D-1,.153430107688651D-1,    WAN 187Ø
* .158Ø37286593993D-1,.16255ØØØ9Ø97852D-1,.166965578Ø15892D-1,     WAN 188Ø
* .1712813542311 14D-1/                                            WAN 189Ø
 DATA W(96), W(97), W(98), W(99), W(1ØØ), W(1Ø1), W(1Ø2),          WAN 19ØØ
* W(1Ø3), W(1Ø4), W(1Ø5), W(1Ø6), W(1Ø7), W(1Ø8), W(1Ø9),           WAN 191Ø
* W(11Ø), W(111), W(112) /.1754947582711 77D-1,                    WAN 192Ø
* .1796Ø327185Ø087D-1,.1836Ø4439373313D-1,.187495869405447D-1,     WAN 193Ø
* .1912752360995Ø9D-1,.19494Ø28Ø587Ø66D-1,.19848881232830 9D-1,    WAN 194Ø
* .2Ø191871Ø4213ØØD-1,.2Ø522792486960 1D-1,.2Ø8414477807511D-1,    WAN 195Ø
* .211476464682213D-1,.214412Ø55392Ø85D-1,.217219495380521D-1,     WAN 196Ø
* .2198971Ø66846Ø5D-1,.222443288937998D-1,.22485652Ø32745ØD-1,    WAN 197Ø
* .227135358502365D-1/                                             WAN 198Ø
 DATA W(113), W(114), W(115), W(116), W(117), W(118), W(119),      WAN 199Ø
* W(12Ø), W(121), W(122), W(123), W(124), W(125), W(126),           WAN 2ØØØ
* W(127) /.229278441436868D-1,.231284488243870D-1,                 WAN 2Ø1Ø
* .233152299940628D-1,.23488Ø76Ø165359D-1,.236468835844476D-1,     WAN 2Ø2Ø
* .237915577810Ø34D-1,.23922Ø121367Ø35D-1,.24Ø38168681Ø241D-1,     WAN 2Ø3Ø
* .241399579890193D-1,.242273192228152D-1,.243ØØ2ØØ1679719D-1,    WAN 2Ø4Ø
* .2435855726469Ø6D-1,.244Ø23556338496D-1,.24431569Ø9785ØØD-1,    WAN 2Ø5Ø
* .2444618Ø1962625D-1/                                             WAN 2Ø6Ø
 DATA W(128), W(129), W(13Ø), W(131), W(132), W(133), W(134),      WAN 2Ø7Ø
* W(135), W(136), W(137), W(138), W(139), W(14Ø), W(141),          WAN 2Ø8Ø
* W(142) /.112789Ø17822272D-3,.262534944296446D-3,                 WAN 2Ø9Ø
* .412463254426176D-3,.56234895403141ØD-3,.712154163473321D-3,     WAN 21ØØ
* .8618537Ø142ØØ89D-3,.1Ø1142439320844D-2,.116Ø84355756772D-2,     WAN 211Ø
* .131ØØ886819Ø25ØD-2,.14591373333107 3D-2,.16Ø796713Ø74933D-2,    WAN 212Ø
* .17565573633Ø73D-2,.19Ø488Ø85349972D-2,.2Ø5292Ø22796614D-2,      WAN 213Ø
* .22ØØ65164983991D-2/                                             WAN 214Ø
 DATA W(143), W(144), W(145), W(146), W(147), W(148), W(149),      WAN 215Ø
* W(15Ø), W(151), W(152), W(153), W(154), W(155), W(156),          WAN 216Ø
* W(157) /.2348Ø5295632731D-2,.24951Ø2Ø347Ø371D-2,                 WAN 217Ø
* .264177682542749D-2,.2788Ø5532532771D-2,.2933915590 82972D-2,    WAN 218Ø
* .3Ø7933574119934D-2,.322429396179420D-2,.3368768Ø731555D-2,     WAN 219Ø
* .3512737705Ø5631D-2,.365617995814250D-2,.3799Ø7374876626D-2,    WAN 22ØØ
* .3941397641Ø883D-2,.4Ø8313Ø286Ø5267D-2,.422425Ø42138154D-2,     WAN 221Ø
* .436473687796806D-2/                                             WAN 222Ø
 DATA W(158), W(159), W(16Ø), W(161), W(162), W(163), W(164),      WAN 223Ø
* W(165), W(166), W(167), W(168), W(169), W(17Ø), W(171),          WAN 224Ø
* W(172) /.45Ø4568581447 90D-2,.4643724555680Ø6D-2,                WAN 225Ø
* .478218392589269D-2,.491992592181387D-2,.5Ø56929880 78684D-2,    WAN 226Ø
* .519317525086928D-2,.532864159391593D-2,.546330858864431D-2,    WAN 227Ø
* .5597156Ø3368291D-2,.573Ø16385Ø6Ø144D-2,.5862312Ø8692265D-2,     WAN 228Ø
* .599358Ø91911534D-2,.612395Ø65556793D-2,.62534Ø173954240D-2,     WAN 229Ø
* .6381914752Ø788D-2/                                              WAN 23ØØ
 DATA W(173), W(174), W(175), W(176), W(177), W(178), W(179),      WAN 231Ø
* W(18Ø), W(181), W(182), W(183), W(184), W(185), W(186),          WAN 232Ø
* W(187) /.65Ø947Ø415Ø5366D-2,.6636Ø4593781Ø7D-2,                  WAN 233Ø
* .6761633ØØ1738ØD-2,.68862Ø269544632D-2,.7ØØ9739Ø9296982D-2,     WAN 234Ø
* .7132223961Ø7539D-2,.725363892583391D-2,.737396577381235D-2,     WAN 235Ø
* .7493186454Ø8588D-2,.7611283Ø8454566D-2,.772823794738156D-2,     WAN 236Ø
* .7844Ø3349893971D-2,.795865236875435D-2,.8Ø72Ø773628735ØD-2,     WAN 237Ø
* .818429146643827D-2/                                             WAN 238Ø
 DATA W(188), W(189), W(19Ø), W(191), W(192), W(193), W(194),      WAN 239Ø
* W(195), W(196), W(197), W(198), W(199), W(2ØØ), W(2Ø1),          WAN 24ØØ
* W(2Ø2) /.829527784623523D-2,.84Ø5Ø1985322154D-2,                WAN 241Ø
* .8513501Ø25Ø2249D-2,.862Ø7Ø5Ø884Ø1Ø1D-2,.872661596169881D-2,    WAN 242Ø
* .883121775724875D-2,.893449478375821D-2,.9Ø3643154866287D-2,     WAN 243Ø
* .9137Ø1276Ø45Ø81D-2,.923622333Ø9563ØD-2,.933Ø4837762327D-2,     WAN 244Ø
* .943Ø47322573775D-2,.952548341Ø62928D-2,.9619Ø6467984Ø73D-2,     WAN 245Ø
* .97112Ø299526628D-2/                                             WAN 246Ø
```

```
      DATA W(2Ø3), W(2Ø4), W(2Ø5), W(2Ø6), W(2Ø7), W(2Ø8), W(2Ø9),      WAN 247Ø
     * W(21Ø), W(211), W(212), W(213), W(214), W(215), W(216),          WAN 248Ø
     * W(217) /.98Ø188453525733D-2,.989109569669583D-2,                 WAN 249Ø
     * .9978823Ø97Ø3491D-2,.1ØØ65Ø535763Ø64D-1,.1Ø149774199Ø949D-1,     WAN 25ØØ
     * .1Ø2329722564782D-1,.1Ø3146352679340D-1,.1Ø39475Ø9832117D-1,     WAN 251Ø
     * .1Ø4733Ø738417Ø4D-1,.1Ø55Ø2926865815D-1,.1Ø6256953418966D-1,     WAN 252Ø
     * .1Ø6995Ø4Ø389798D-1,.1Ø7717Ø77Ø58Ø46D-1,.1Ø8422955111148D-1,     WAN 253Ø
     * .1Ø9112568660490D-1/                                             WAN 254Ø
      DATA W(218), W(219), W(22Ø), W(221), W(222), W(223), W(224),      WAN 255Ø
     * W(225), W(226), W(227), W(228), W(229), W(23Ø), W(231),          WAN 256Ø
     * W(232) /.1Ø9785814257296D-1,.11Ø44259Ø9Ø8139D-1,                 WAN 257Ø
     * .111Ø828ØØØ9Ø098D-1,.1117Ø6345765534D-1,.112313134396497D-1,     WAN 258Ø
     * .1129Ø3Ø74958755D-1,.113476Ø78955455D-1,.114Ø32Ø6Ø43Ø392D-1,     WAN 259Ø
     * .11457Ø93598Ø9Ø6D-1,.115Ø9262477Ø395D-1,.115597Ø4854Ø436D-1,     WAN 26ØØ
     * .116Ø84131622531D-1,.1165538ØØ949452D-1,.117ØØ5986Ø662Ø7D-1,     WAN 261Ø
     * .11744Ø6191406Ø6D-1/                                             WAN 262Ø
      DATA W(233), W(234), W(235), W(236), W(237), W(238), W(239),      WAN 263Ø
     * W(24Ø), W(241), W(242), W(243), W(244), W(245), W(246),          WAN 264Ø
     * W(247) /.117857634973434D-1,.1182569710Ø824ØD-1,                 WAN 265Ø
     * .118638567340711D-1,.119ØØ2366727665D-1,.119348314595636D-1,     WAN 266Ø
     * .119676359Ø49Ø59D-1,.11998645Ø878Ø58D-1,.12Ø2785435658 26D-1,    WAN 267Ø
     * .12Ø5525932956Ø1D-1,.12Ø8Ø8558957245D-1,.121Ø464Ø2153Ø5D-1,     WAN 268Ø
     * .121266Ø872Ø5273D-1,.121467581157945D-1,.121650853785355D-1,     WAN 269Ø
     * .121815877594818D-1/                                             WAN 27ØØ
      DATA W(248), W(249), W(25Ø), W(251), W(252), W(253), W(254),      WAN 271Ø
     * W(255) /.121962627831147D-1,.122Ø91Ø8248Ø372D-1,                 WAN 272Ø
     * .1222Ø1222273Ø4ØD-1,.122293Ø3Ø6871Ø3D-1,.12236649395Ø4Ø2D-1,     WAN 273Ø
     * .1224216Ø1Ø42728D-1,.122458343697479D-1,.12247671640 2898D-1/    WAN 274Ø
      LOOP = MAXØ(1,N/256)                                              WAN 275Ø
      FLOOP = LOOP                                                      WAN 276Ø
      H = (B-A)/FLOOP                                                   WAN 277Ø
      SCALE = H/2.ØDØ                                                  WAN 278Ø
      M = MINØ(128,N/2)                                                 WAN 279Ø
      MT = 2*M                                                          WAN 28ØØ
      NPLACE = M - 1                                                    WAN 281Ø
      DO 2Ø L=1,LOOP                                                    WAN 282Ø
        FL = L                                                          WAN 283Ø
        AL = A + (FL-1.ØDØ)*H                                           WAN 284Ø
        BL = A + FL*H                                                   WAN 285Ø
        K = 256*(L-1)                                                   WAN 286Ø
        DO 1Ø I=1,M                                                     WAN 287Ø
          NPI = NPLACE + I                                              WAN 288Ø
          S = T(NPI)                                                    WAN 289Ø
          R = W(NPI)*SCALE                                              WAN 29ØØ
          I1 = K + I                                                    WAN 291Ø
          I2 = K + MT + 1 - I                                           WAN 292Ø
          TV(I1) = (AL*(1.DØ+S)+(1.ØDØ-S)*BL)/2.DØ                      WAN 293Ø
          TV(I2) = (AL*(1.DØ-S)+(1.ØDØ+S)*BL)/2.DØ                      WAN 294Ø
          WV(I1) = R                                                    WAN 295Ø
          WV(I2) = R                                                    WAN 296Ø
   1Ø   CONTINUE                                                        WAN 297Ø
   2Ø CONTINUE                                                          WAN 298Ø
      RETURN                                                            WAN 299Ø
      END                                                               WAN 3ØØØ


      DOUBLE PRECISION FUNCTION RMIN(N, M, COND, UNITRD, AVERR)         RMI   1Ø
C FOR A LINEAR SYSTEM   (I-KMM)*XM=RHFCN   OF ORDER M, THIS IS THE      RMI   2Ø
C VALUE OF RELMIN USED IN IEGS. THE VARIABLE UNITRD IS DEFINED IN       RMI   3Ø
C IEGAUS, AND THE VARIABLES COND AND AVERR ARE DEFINED IN IEGS          RMI   4Ø
C USING CONEW.                                                          RMI   5Ø
C IT IS UNLIKELY THAT A SOLUTION X CAN BE FOUND FOR THE ORIGINAL        RMI   6Ø
C INTEGRAL EQUATION WITH A SMALLER RELATIVE ERROR THAN RMIN.            RMI   7Ø
      DOUBLE PRECISION FLOAT1, FLOAT2, COND, AVERR, UNITRD, DMAX1       RMI   8Ø
      FLOAT1 = M                                                        RMI   9Ø
      FLOAT2 = M/N                                                      RMI  1ØØ
      RMIN = DMAX1((FLOAT1**1.5DØ)*COND*UNITRD,(FLOAT2**1.5DØ)*         RMI  11Ø
     * AVERR)                                                           RMI  12Ø
      RETURN                                                            RMI  13Ø
      END                                                               RMI  14Ø


      DOUBLE PRECISION FUNCTION CONEW(COND, ELINSY, RELRSD, AVERR,      CON   1Ø
     * PASTC, PASTRE)                                                   CON   2Ø
```

```
C THIS IS USED IN UPDATING THE VALUE OF THE CONDITION NUMBER          CON   3Ø
C IN IEGS.                                                            CON   4Ø
      DOUBLE PRECISION COND, ELINSY, RELRSD, AVERR, PASTC, PASTRE,    CON   5Ø
     * C, DSQRT, DMAX1                                               CON   6Ø
      AVERR = DSQRT(ELINSY*PASTRE)                                    CON   7Ø
      PASTRE = ELINSY                                                 CON   8Ø
      IF (RELRSD.EQ.Ø.ØDØ) GO TO 1Ø                                   CON   9Ø
      C = DMAX1(1.ØDØ,ELINSY/RELRSD)                                  CON  1ØØ
      CONEW = DSQRT(C*PASTC)                                          CON  11Ø
      PASTC = C                                                       CON  12Ø
      RETURN                                                          CON  13Ø
   1Ø CONEW = COND                                                    CON  14Ø
      RETURN                                                          CON  15Ø
      END                                                             CON  16Ø



      SUBROUTINE LEAVE(IERSET, NF, MF, XV, TV, WV, ERROR, KERNEL,     LEA   1Ø
     * RHFCN, EP, IFLAG, X, T, NT, IER, EPS, ELINSY, TN, WN, TM,      LEA   2Ø
     * WM, XM, XMZ, KMM, KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN,   LEA   3Ø
     * NUP, NHALF, XNORM)                                             LEA   4Ø
C THIS ROUTINE SETS ALL NECESSARY PARAMETERS FOR LEAVING IEGAUS.      LEA   5Ø
C IF NT .GT. Ø, IT ALSO PERFORMS THE NECESSARY NYSTROM                LEA   6Ø
C INTERPOLATION AT THE NODES GIVEN IN T.                              LEA   7Ø
      DOUBLE PRECISION KERNEL, RHFCN, EP, X, T, XV, TV, WV, ERROR,    LEA   8Ø
     * EPS, ELINSY, TN, WN, TM, WM, XM, XMZ, KMM, KMN, KNM, RHS,      LEA   9Ø
     * IMKNN, LUFACT, R, RH, DELN, NORMK, R1, R2, FINLEP, SAVEP,      LEA  1ØØ
     * SUM, DERROR, NUMR1, NORM, TEMP, DABS, DMAX1, XNORM             LEA  11Ø
      DIMENSION X(1), T(1), XV(MF), TV(MF), WV(MF), TN(NF), WN(NF),   LEA  12Ø
     * WM(MF), XM(MF), XMZ(MF), KMM(NUP,NUP), KMN(NUP,NHALF),         LEA  13Ø
     * KNM(NHALF,NUP), RHS(MF), IMKNN(NUP,NUP), LUFACT(NUP,NUP),      LEA  14Ø
     * R(MF), RH(NF), TM(MF), DELN(NF)                                LEA  15Ø
      COMMON /INFO/ R1, R2, FINLEP, NORMK, NFINAL, MFINAL             LEA  16Ø
      EXTERNAL KERNEL, RHFCN                                          LEA  17Ø
C SET ERROR PARAMETERS FOR RETURN.                                    LEA  18Ø
      NORMK = Ø.ØDØ                                                   LEA  19Ø
      NFINAL = NF                                                     LEA  2ØØ
      MFINAL = MF                                                     LEA  21Ø
      FINLEP = EPS                                                    LEA  22Ø
      IF ((EPS.GT.EP) .AND. (ERROR.LE.EPS)) GO TO 1Ø                  LEA  23Ø
      IER = IERSET                                                    LEA  24Ø
      EP = ERROR                                                      LEA  25Ø
      IF (NT.EQ.Ø) GO TO 2Ø                                           LEA  26Ø
      GO TO 4Ø                                                        LEA  27Ø
   1Ø IER = 3                                                         LEA  28Ø
C SINCE EPS IS THE SMALLEST ERROR POSSIBLE, SET EP=EPS FOR THE        LEA  29Ø
C RETURN ERROR ESTIMATE.                                              LEA  3ØØ
      EP = EPS                                                        LEA  31Ø
      IF (NT.GT.Ø) GO TO 4Ø                                           LEA  32Ø
C NO NYSTROM INTERPOLATION IS DESIRED. RETURN THE VALUES AT THE       LEA  33Ø
C GAUSSIAN NODE POINTS.                                               LEA  34Ø
   2Ø DO 3Ø I=1,MF                                                    LEA  35Ø
        X(I) = XV(I)                                                  LEA  36Ø
        T(I) = TV(I)                                                  LEA  37Ø
   3Ø CONTINUE                                                        LEA  38Ø
      NT = MF                                                         LEA  39Ø
      RETURN                                                          LEA  4ØØ
C CALCULATE NORM(K).                                                  LEA  41Ø
   4Ø SAVEP = EP                                                      LEA  42Ø
      DO 5Ø I=1,NF                                                    LEA  43Ø
        IMKNN(I,I) = IMKNN(I,I) - 1.ØDØ                               LEA  44Ø
   5Ø CONTINUE                                                        LEA  45Ø
      NORMK = Ø.ØDØ                                                   LEA  46Ø
      DO 7Ø I=1,NF                                                    LEA  47Ø
        SUM = Ø.ØDØ                                                   LEA  48Ø
        DO 6Ø J=1,NF                                                  LEA  49Ø
          SUM = SUM + DABS(IMKNN(I,J))                                LEA  5ØØ
   6Ø   CONTINUE                                                      LEA  51Ø
        NORMK = DMAX1(NORMK,SUM)                                      LEA  52Ø
   7Ø CONTINUE                                                        LEA  53Ø
      DO 8Ø I=1,NF                                                    LEA  54Ø
        IMKNN(I,I) = IMKNN(I,I) + 1.ØDØ                               LEA  55Ø
   8Ø CONTINUE                                                        LEA  56Ø
      IF (NF.EQ.MF) GO TO 13Ø                                         LEA  57Ø
C ITERATE TO DECREASE THE NOISE LEVEL IN X. THIS SHOULD REDUCE        LEA  58Ø
C POSSIBLE ERRORS IN NYSTROM INTERPOLATION.                           LEA  59Ø
```

```
      DERROR = ((1.ØDØ-R1)/R1)*EPS/NORMK                                LEA  6ØØ
      IF (IFLAG.EQ.1) DERROR = DERROR*XNORM                             LEA  61Ø
      ITLOOP = Ø                                                        LEA  62Ø
      DO 9Ø I=1,MF                                                      LEA  63Ø
        XM(I) = XV(I)                                                   LEA  64Ø
   9Ø CONTINUE                                                          LEA  65Ø
  1ØØ DO 11Ø I=1,MF                                                     LEA  66Ø
        XMZ(I) = XM(I)                                                  LEA  67Ø
  11Ø CONTINUE                                                          LEA  68Ø
      CALL ITERT(KERNEL, RHFCN, NF, TN, WN, MF, TM, WM, XM, XMZ,        LEA  69Ø
     * KMM, KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN, NUP, NHALF,     LEA  7ØØ
     * 1)                                                               LEA  71Ø
      NUMR1 = NORM(XM,XMZ,MF,1)                                         LEA  72Ø
      ITLOOP = ITLOOP + 1                                               LEA  73Ø
      IF ((NUMR1.GT.DERROR) .AND. (ITLOOP.LT.5)) GO TO 1ØØ              LEA  74Ø
      DO 12Ø I=1,MF                                                     LEA  75Ø
        XV(I) = XM(I)                                                   LEA  76Ø
  12Ø CONTINUE                                                          LEA  77Ø
C ESTIMATE NEW ERROR BOUND FOR NYSTROM INTERPOLATES.                    LEA  78Ø
      TEMP = NORMK*(R1/(1.ØDØ-R1))*NUMR1                                LEA  79Ø
      IF (IFLAG.EQ.1) TEMP = TEMP/XNORM                                 LEA  8ØØ
      EP = DMAX1(EP,TEMP)                                               LEA  81Ø
      GO TO 14Ø                                                         LEA  82Ø
C NO ITERATION USED IN COMPUTING X. JUST COMPUTE ERROR ESTIMATE IN      LEA  83Ø
C NYSTROM INTERPOLATE.                                                  LEA  84Ø
  13Ø TEMP = NORMK*ELINSY                                               LEA  85Ø
      IF (IFLAG.EQ.Ø) TEMP = TEMP*XNORM                                 LEA  86Ø
      IF (IER.NE.2) EP = DMAX1(EP,TEMP)                                 LEA  87Ø
C COMPUTE NYSTROM INTERPOLATES AT THE NODES IN T.                       LEA  88Ø
  14Ø DO 16Ø I=1,NT                                                     LEA  89Ø
        SUM = Ø.ØDØ                                                     LEA  9ØØ
        DO 15Ø J=1,MF                                                   LEA  91Ø
          SUM = SUM + WV(J)*KERNEL(T(I),TV(J))*XV(J)                    LEA  92Ø
  15Ø   CONTINUE                                                        LEA  93Ø
        X(I) = RHFCN(T(I)) + SUM                                        LEA  94Ø
  16Ø CONTINUE                                                          LEA  95Ø
      IF ((IER.EQ.Ø) .AND. (EP.GT.EPS)) IER = 4                         LEA  96Ø
      IF ((IER.EQ.1) .AND. (EP.GT.ERROR)) IER = 5                       LEA  97Ø
      IF ((IER.EQ.3) .AND. (EP.GT.EPS)) IER = 6                         LEA  98Ø
      EP = SAVEP                                                        LEA  99Ø
      RETURN                                                            LEA 1ØØØ
      END                                                               LEA 1Ø1Ø
```

```
      SUBROUTINE INTERP(TM, WM, XM, M, TN, WN, XN, N, KERNEL,           INT   1Ø
     * RHFCN, RHS, KMN, NHALF, NUP)                                     INT   2Ø
C USE THE VALUES OF XN(I), I=1,...,N, TO CALCULATE THE NYSTROM          INT   3Ø
C INTERPOLATES XM(I), I=1,...,M.                                        INT   4Ø
      DOUBLE PRECISION KERNEL, RHFCN, TM, WM, XM, TN, WN, XN, RHS,      INT   5Ø
     * KMN                                                             INT   6Ø
      DIMENSION TM(M), WM(M), XM(M), TN(N), WN(N), XN(N), RHS(M),       INT   7Ø
     * KMN(NUP,NHALF)                                                  INT   8Ø
      IF (M.GT.NUP) GO TO 6Ø                                            INT   9Ø
C SINCE M .LE. NUPPER, SAVE K(TM(I),TN(J))=KMN(I,J) AND                 INT  1ØØ
C RHS(I)=RHFCN(TM(I)) FOR LATER USE IN ITERT.                           INT  11Ø
      DO 2Ø I=1,M                                                       INT  12Ø
        DO 1Ø J=1,N                                                     INT  13Ø
          KMN(I,J) = WN(J)*KERNEL(TM(I),TN(J))                          INT  14Ø
   1Ø   CONTINUE                                                        INT  15Ø
   2Ø CONTINUE                                                          INT  16Ø
      DO 3Ø I=1,M                                                       INT  17Ø
        RHS(I) = RHFCN(TM(I))                                           INT  18Ø
        XM(I) = RHS(I)                                                  INT  19Ø
   3Ø CONTINUE                                                          INT  2ØØ
C CALCULATE NYSTROM INTERPOLATING FORMULA.                              INT  21Ø
      DO 5Ø I=1,M                                                       INT  22Ø
        DO 4Ø J=1,N                                                     INT  23Ø
          XM(I) = XM(I) + KMN(I,J)*XN(J)                                INT  24Ø
   4Ø   CONTINUE                                                        INT  25Ø
   5Ø CONTINUE                                                          INT  26Ø
      RETURN                                                            INT  27Ø
C M .GT. NUPPER, SO SAVE JUST RHS(I) FOR LATER USE IN ITERT.            INT  28Ø
C CALCULATE NYSTROM INTERPOLATING FORMULA.                              INT  29Ø
   6Ø DO 8Ø I=1,M                                                       INT  3ØØ
        RHS(I) = RHFCN(TM(I))                                           INT  31Ø
```

```
      XM(I) = RHS(I)                                                    INT  32Ø
      DO 7Ø J=1,N                                                       INT  33Ø
        XM(I) = XM(I) + WN(J)*KERNEL(TM(I),TN(J))*XN(J)                 INT  34Ø
 7Ø   CONTINUE                                                          INT  35Ø
 8Ø CONTINUE                                                            INT  36Ø
      RETURN                                                            INT  37Ø
      END                                                               INT  38Ø



      SUBROUTINE ITERT(KERNEL, RHFCN, N, TN, WN, M, TM, WM, XM,         ITE   1Ø
     * XMZ, KMM, KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN, NUP,       ITE   2Ø
     * NHALF, IFLG)                                                    ITE   3Ø
C THIS ROUTINE CALCULATES ONE ITERATE XM GIVEN THE INITIAL GUESS        ITE   4Ø
C XMZ. THE ROUTINE IS DIVIDED ACCORDING TO WHETHER OR NOT               ITE   5Ø
C M .GT. NUPPER.                                                        ITE   6Ø
      DOUBLE PRECISION KERNEL, RHFCN, TN, WN, TM, WM, XM, XMZ, KMM,     ITE   7Ø
     * KMN, KNM, RHS, IMKNN, LUFACT, R, RH, DELN, SUM, DET             ITE   8Ø
      DIMENSION TN(N), WN(N), TM(M), WM(M), XM(M), XMZ(M),              ITE   9Ø
     * KMM(NUP,NUP), KMN(NUP,NHALF), KNM(NHALF,NUP), RHS(M),           ITE  1ØØ
     * IMKNN(NUP,NUP), LUFACT(NUP,NUP), R(M), RH(N), DELN(N)           ITE  11Ø
C M .GT. NUPPER MEANS THAT THE MATRICES KMM,KMN,KNM CAN NO LONGER       ITE  12Ø
C BE STORED DUE TO LACK OF SPACE.                                       ITE  13Ø
      IF (M.GT.NUP) GO TO 12Ø                                           ITE  14Ø
      IF (IFLG.EQ.1) GO TO 4Ø                                           ITE  15Ø
C IF IFLG=Ø, THEN THE MATRICES KMM AND KNM MUST BE COMPUTED             ITE  16Ø
C AND STORED.                                                           ITE  17Ø
      DO 3Ø J=1,M                                                       ITE  18Ø
        DO 1Ø I=1,M                                                     ITE  19Ø
          KMM(I,J) = WM(J)*KERNEL(TM(I),TM(J))                         ITE  2ØØ
 1Ø   CONTINUE                                                          ITE  21Ø
        DO 2Ø I=1,N                                                     ITE  22Ø
          KNM(I,J) = WM(J)*KERNEL(TN(I),TM(J))                         ITE  23Ø
 2Ø   CONTINUE                                                          ITE  24Ø
 3Ø CONTINUE                                                            ITE  25Ø
C COMPUTE  RESIDUALS R(I)=RHFCN(TM(I))-XMZ(I)+KM(TM(I))*XMZ(I)          ITE  26Ø
 4Ø DO 6Ø I=1,M                                                         ITE  27Ø
      SUM = Ø.ØDØ                                                       ITE  28Ø
      DO 5Ø J=1,M                                                       ITE  29Ø
        SUM = SUM + KMM(I,J)*XMZ(J)                                     ITE  3ØØ
 5Ø   CONTINUE                                                          ITE  31Ø
      R(I) = RHS(I) - (XMZ(I)-SUM)                                      ITE  32Ø
 6Ø CONTINUE                                                            ITE  33Ø
C COMPUTE RH=KM*R AT ALL TN(I).                                         ITE  34Ø
      DO 8Ø I=1,N                                                       ITE  35Ø
        RH(I) = Ø.ØDØ                                                   ITE  36Ø
        DO 7Ø J=1,M                                                     ITE  37Ø
          RH(I) = RH(I) + KNM(I,J)*R(J)                                 ITE  38Ø
 7Ø   CONTINUE                                                          ITE  39Ø
 8Ø CONTINUE                                                            ITE  4ØØ
C    CALCULATE DELN=((I-KN)**(-1))*KM*R AT ALL TN(I).                   ITE  41Ø
C ***********************************************************************ITE  42Ø
C                                                                      *ITE  43Ø
      CALL LINSYS(IMKNN, LUFACT, N, RH, DELN, 4, DET, NUP)              ITE  44Ø
C                                                                      *ITE  45Ø
C    SEE THE ORIGINAL REFERENCE IN IEGS.                               *ITE  46Ø
C ***********************************************************************ITE  47Ø
C    CALCULATE NEW XM.                                                 ITE  48Ø
      DO 11Ø I=1,M                                                      ITE  49Ø
        SUM = Ø.ØDØ                                                     ITE  5ØØ
        DO 9Ø J=1,M                                                     ITE  51Ø
          SUM = SUM + KMM(I,J)*R(J)                                     ITE  52Ø
 9Ø   CONTINUE                                                          ITE  53Ø
        DO 1ØØ J=1,N                                                    ITE  54Ø
          SUM = SUM + KMN(I,J)*DELN(J)                                  ITE  55Ø
 1ØØ  CONTINUE                                                          ITE  56Ø
      XM(I) = SUM + R(I) + XMZ(I)                                       ITE  57Ø
 11Ø CONTINUE                                                           ITE  58Ø
      RETURN                                                            ITE  59Ø
C ENTRANCE WHEN M .GT. NUP.                                             ITE  6ØØ
C CALCULATE RESIDUALS.                                                  ITE  61Ø
 12Ø DO 14Ø I=1,M                                                       ITE  62Ø
      SUM = Ø.ØDØ                                                       ITE  63Ø
      DO 13Ø J=1,M                                                      ITE  64Ø
        SUM = SUM + WM(J)*KERNEL(TM(I),TM(J))*XMZ(J)                    ITE  65Ø
 13Ø  CONTINUE                                                          ITE  66Ø
```

```
      R(I) = RHS(I) - (XMZ(I)-SUM)                                    ITE  67Ø
  14Ø CONTINUE                                                        ITE  68Ø
C CALCULATE RH=KM*R.                                                  ITE  69Ø
      DO 16Ø I=1,N                                                    ITE  7ØØ
        RH(I) = Ø.ØDØ                                                 ITE  71Ø
        DO 15Ø J=1,M                                                  ITE  72Ø
          RH(I) = RH(I) + WM(J)*KERNEL(TN(I),TM(J))*R(J)              ITE  73Ø
  15Ø   CONTINUE                                                      ITE  74Ø
  16Ø CONTINUE                                                        ITE  75Ø
C ********************************************************************  ITE  76Ø
C                                                                   *  ITE  77Ø
      CALL LINSYS(IMKNN, LUFACT, N, RH, DELN, 4, DET, NUP)            ITE  78Ø
C                                                                   *  ITE  79Ø
C    SEE THE ORIGINAL REFERENCE IN IEGS.                            *  ITE  8ØØ
C ********************************************************************  ITE  81Ø
C    CALCULATE XM.                                                    ITE  82Ø
      DO 19Ø I=1,M                                                    ITE  83Ø
        SUM = Ø.ØDØ                                                   ITE  84Ø
        DO 17Ø J=1,M                                                  ITE  85Ø
          SUM = SUM + WM(J)*KERNEL(TM(I),TM(J))*R(J)                  ITE  86Ø
  17Ø   CONTINUE                                                      ITE  87Ø
        DO 18Ø J=1,N                                                  ITE  88Ø
          SUM = SUM + WN(J)*KERNEL(TM(I),TN(J))*DELN(J)               ITE  89Ø
  18Ø   CONTINUE                                                      ITE  9ØØ
        XM(I) = SUM + R(I) + XMZ(I)                                   ITE  91Ø
  19Ø CONTINUE                                                        ITE  92Ø
      RETURN                                                          ITE  93Ø
      END                                                             ITE  94Ø


      DOUBLE PRECISION FUNCTION NORM(X, Y, N, IFLAG)                  NOR   1Ø
C IFLAG=Ø    CALCULATE THE MAXIMUM NORM OF X.                         NOR   2Ø
C IFLAG=1    CALCULATE THE MAXIMUM NORM OF X-Y.                       NOR   3Ø
      DOUBLE PRECISION X, Y, DMAX1, DABS                              NOR   4Ø
      DIMENSION X(N), Y(N)                                            NOR   5Ø
      IF (IFLAG.EQ.1) GO TO 2Ø                                        NOR   6Ø
C FIND THE NORM OF X.                                                 NOR   7Ø
      NORM = Ø.ØDØ                                                   NOR   8Ø
      DO 1Ø I=1,N                                                     NOR   9Ø
        NORM = DMAX1(NORM,DABS(X(I)))                                 NOR  1ØØ
   1Ø CONTINUE                                                        NOR  11Ø
      RETURN                                                          NOR  12Ø
C FIND THE NORM OF X-Y.                                               NOR  13Ø
   2Ø NORM = Ø.ØDØ                                                    NOR  14Ø
      DO 3Ø I=1,N                                                     NOR  15Ø
        NORM = DMAX1(NORM,DABS(X(I)-Y(I)))                            NOR  16Ø
   3Ø CONTINUE                                                        NOR  17Ø
      RETURN                                                          NOR  18Ø
      END                                                             NOR  19Ø


      SUBROUTINE LINSYS(A, D, N, B, X, OPTION, DET, MACHIN)           LIN   1Ø
C SOLVE A*X=B, ORDER(A)=N, DIMENSION OF A=MACHIN.                     LIN   2Ø
C OPTION=1   SOLVE A*X=B, LEAVE THE LU DECOMPOSITION A=L*U IN D       LIN   3Ø
C            AND THE PIVOTS IN PIVOT. THE ANSWERS ARE LEFT IN X.      LIN   4Ø
C            IT IS PERMISSABLE TO LET B=X AND D=A, BUT THEN THE       LIN   5Ø
C            ORIGINAL CONTENTS OF A AND B ARE LOST.                   LIN   6Ø
C OPTION=2   CALCULATE DECOMPOSITION A=L*U INCLUDING PIVOTS. SOLVE    LIN   7Ø
C            A*X=B, AND THEN CALCULATE THE RESIDUAL AND ONE           LIN   8Ø
C            CORRECTION. THE CORRECTIONS ARE LEFT IN R, THE NEW       LIN   9Ø
C            VALUE OF X1 IN X, THE RELATIVE ERROR                     LIN  1ØØ
C                      NORM(XØ-X1)/NORM(X1)                           LIN  11Ø
C            IN THE VARIABLE ERROR, AND THE RELATIVE RESIDUAL         LIN  12Ø
C                      NORM(RESIDUAL)/NORM(B)                         LIN  13Ø
C            IN THE VARIABLE RELRSD. THESE VALUES CAN BE OBTAINED     LIN  14Ø
C            USING THE COMMON/LINEAR/ GIVEN BELOW.                    LIN  15Ø
C OPTION=3   SAME AS OPTION=1, EXCEPT A=L*U IS KNOWN AND STORED       LIN  16Ø
C            IN D.                                                    LIN  17Ø
C OPTION=4   SAME AS OPTION=2, EXCEPT A=L*U IS KNOWN AND STORED       LIN  18Ø
C            IN D.                                                    LIN  19Ø
C THE DECOMPOSITION OF A INTO L*U USES SCALED PARTIAL PIVOTING IN     LIN  2ØØ
C THE COLUMNS. FOR OPTIONS 1 AND 2, THE DETERMINANT OF A IS           LIN  21Ø
C CALCULATED AND STORED IN DET. IF DET=Ø, THEN THE ANSWERS ARE        LIN  22Ø
C NONSENSE, D AND X DO NOT CONTAIN USEFUL INFORMATION, AND AND A      LIN  23Ø
```

```
C AND B ARE LEFT UNDISTURBED (UNLESS D=A OR X=B).                    LIN  24Ø
C LINSYS IS PRESENTLY LIMITED TO N .LE. 1ØØ. TO REMOVE THIS          LIN  25Ø
C RESTRICTION,CHANGE THE DIMENSION STATEMENT FOR THE ARRAYS R,       LIN  26Ø
C SCALE, AND PIVOT, WHICH ARE GIVEN IN COMMON/LINEAR/.               LIN  27Ø
      INTEGER OPTION, PIVOT                                          LIN  28Ø
      DOUBLE PRECISION NORMX, NORME, NORMB, NORMR, A, D, B, X, R,    LIN  29Ø
     * SCALE, ERROR, RELRSD, DET, C, TEMP, DMAX1, DABS, SUM          LIN  3ØØ
      DIMENSION A(MACHIN,MACHIN), D(MACHIN,MACHIN), B(N), X(N)       LIN  31Ø
      COMMON /LINEAR/ R(1ØØ), SCALE(1ØØ), ERROR, RELRSD, PIVOT(1ØØ) LIN  32Ø
      ISWIT = 1                                                      LIN  33Ø
      IF (OPTION.GT.2) GO TO 11Ø                                     LIN  34Ø
C PRODUCE LU DECOMPOSITION OF A AND DET(A)                           LIN  35Ø
      DET = Ø.ØDØ                                                    LIN  36Ø
      DO 2Ø I=1,N                                                    LIN  37Ø
        DO 1Ø J=1,N                                                  LIN  38Ø
          D(I,J) = A(I,J)                                            LIN  39Ø
   1Ø   CONTINUE                                                     LIN  4ØØ
   2Ø CONTINUE                                                       LIN  41Ø
C PRODUCE SCALING FACTORS FOR SCALED PARTIAL PIVOTING.               LIN  42Ø
      DO 4Ø I=1,N                                                    LIN  43Ø
        SCALE(I) = Ø.ØDØ                                             LIN  44Ø
        DO 3Ø J=1,N                                                  LIN  45Ø
          SCALE(I) = DMAX1(SCALE(I),DABS(D(I,J)))                    LIN  46Ø
   3Ø   CONTINUE                                                     LIN  47Ø
        IF (SCALE(I).EQ.Ø.ØDØ) RETURN                                LIN  48Ø
   4Ø CONTINUE                                                       LIN  49Ø
      DET = 1.ØDØ                                                    LIN  5ØØ
      NM1 = N - 1                                                    LIN  51Ø
C BEGIN GAUSSIAN ELIMINATION.                                        LIN  52Ø
      DO 1ØØ K=1,NM1                                                 LIN  53Ø
C SELECT PIVOT ROW.                                                  LIN  54Ø
        C = DABS(D(K,K))/SCALE(K)                                    LIN  55Ø
        INDEX = K                                                    LIN  56Ø
        KP1 = K + 1                                                  LIN  57Ø
        DO 5Ø I=KP1,N                                                LIN  58Ø
          TEMP = DABS(D(I,K))/SCALE(I)                               LIN  59Ø
          IF (TEMP.LE.C) GO TO 5Ø                                    LIN  6ØØ
          INDEX = I                                                  LIN  61Ø
          C = TEMP                                                   LIN  62Ø
   5Ø   CONTINUE                                                     LIN  63Ø
        PIVOT(K) = INDEX                                             LIN  64Ø
        IF (INDEX.EQ.K) GO TO 7Ø                                     LIN  65Ø
C SWITCH ROWS OF D.                                                  LIN  66Ø
        DO 6Ø J=K,N                                                  LIN  67Ø
          TEMP = D(K,J)                                              LIN  68Ø
          D(K,J) = D(INDEX,J)                                        LIN  69Ø
          D(INDEX,J) = TEMP                                          LIN  7ØØ
   6Ø   CONTINUE                                                     LIN  71Ø
        TEMP = SCALE(K)                                              LIN  72Ø
        SCALE(K) = SCALE(INDEX)                                      LIN  73Ø
        SCALE(INDEX) = TEMP                                          LIN  74Ø
        DET = -DET                                                   LIN  75Ø
   7Ø   DET = DET*D(K,K)                                             LIN  76Ø
C ELIMINATE UNKNOWN =K FROM BELOW DIAGONAL IN COLUMN K.              LIN  77Ø
        IF (DET.EQ.Ø.ØDØ) RETURN                                     LIN  78Ø
        DO 9Ø I=KP1,N                                                LIN  79Ø
          D(I,K) = D(I,K)/D(K,K)                                     LIN  8ØØ
          TEMP = D(I,K)                                              LIN  81Ø
          DO 8Ø J=KP1,N                                              LIN  82Ø
            D(I,J) = D(I,J) - TEMP*D(K,J)                            LIN  83Ø
   8Ø     CONTINUE                                                   LIN  84Ø
   9Ø   CONTINUE                                                     LIN  85Ø
  1ØØ CONTINUE                                                       LIN  86Ø
      DET = DET*D(N,N)                                               LIN  87Ø
      IF (DET.EQ.Ø.ØDØ) RETURN                                       LIN  88Ø
C THE DECOMPOSITION A=P*L*U IS COMPLETE.                             LIN  89Ø
C BEGIN SOLUTION OF LINEAR SYSTEM P*L*U*X=B                          LIN  9ØØ
C SET R AND X FOR SOLUTION OF A*X=B.                                 LIN  91Ø
  11Ø DO 12Ø I=1,N                                                   LIN  92Ø
        R(I) = B(I)                                                  LIN  93Ø
        X(I) = Ø.ØDØ                                                 LIN  94Ø
  12Ø CONTINUE                                                       LIN  95Ø
      GO TO 17Ø                                                      LIN  96Ø
C SET R=B-A*X FOR COMPUTATION OF CORRECTION.                         LIN  97Ø
  13Ø DO 15Ø I=1,N                                                   LIN  98Ø
        SUM = Ø.ØDØ                                                  LIN  99Ø
```

```
      DO 14Ø J=1,N                                                    LIN 1ØØØ
        SUM = SUM + A(I,J)*X(J)                                       LIN 1Ø1Ø
14Ø   CONTINUE                                                        LIN 1Ø2Ø
      R(I) = B(I) - SUM                                               LIN 1Ø3Ø
15Ø CONTINUE                                                          LIN 1Ø4Ø
    NORMB = Ø.ØDØ                                                     LIN 1Ø5Ø
    NORMR = Ø.ØDØ                                                     LIN 1Ø6Ø
    DO 16Ø I=1,N                                                      LIN 1Ø7Ø
      NORMB = DMAX1(NORMB,DABS(B(I)))                                 LIN 1Ø8Ø
      NORMR = DMAX1(NORMR,DABS(R(I)))                                 LIN 1Ø9Ø
16Ø CONTINUE                                                          LIN 11ØØ
    RELRSD = NORMR/NORMB                                              LIN 111Ø
    ISWIT = 2                                                         LIN 112Ø
C SOLVE P*L*Z=R AND STORE IN R.                                       LIN 113Ø
17Ø NM1 = N - 1                                                       LIN 114Ø
    DO 2ØØ K=1,NM1                                                    LIN 115Ø
      IF (PIVOT(K).EQ.K) GO TO 18Ø                                    LIN 116Ø
      KPIV = PIVOT(K)                                                 LIN 117Ø
      TEMP = R(K)                                                     LIN 118Ø
      R(K) = R(KPIV)                                                  LIN 119Ø
      R(KPIV) = TEMP                                                  LIN 12ØØ
18Ø   KP1 = K + 1                                                     LIN 121Ø
      DO 19Ø I=KP1,N                                                  LIN 122Ø
        R(I) = R(I) - D(I,K)*R(K)                                     LIN 123Ø
19Ø   CONTINUE                                                        LIN 124Ø
2ØØ CONTINUE                                                          LIN 125Ø
C SOLVE U*E=R AND STORE IN R. ALSO LET X=X+E.                         LIN 126Ø
    R(N) = R(N)/D(N,N)                                                LIN 127Ø
    GO TO (21Ø, 22Ø, 21Ø, 22Ø), OPTION                                LIN 128Ø
21Ø X(N) = R(N)                                                       LIN 129Ø
    GO TO 23Ø                                                         LIN 13ØØ
22Ø X(N) = X(N) + R(N)                                                LIN 131Ø
23Ø DO 27Ø II=1,NM1                                                   LIN 132Ø
      I = N - II                                                      LIN 133Ø
      SUM = Ø.ØDØ                                                     LIN 134Ø
      IP1 = I + 1                                                     LIN 135Ø
      DO 24Ø J=IP1,N                                                  LIN 136Ø
        SUM = SUM + D(I,J)*R(J)                                       LIN 137Ø
24Ø   CONTINUE                                                        LIN 138Ø
      R(I) = (R(I)-SUM)/D(I,I)                                        LIN 139Ø
      GO TO (25Ø, 26Ø, 25Ø, 26Ø), OPTION                              LIN 14ØØ
25Ø   X(I) = R(I)                                                     LIN 141Ø
      GO TO 27Ø                                                       LIN 142Ø
26Ø   X(I) = X(I) + R(I)                                              LIN 143Ø
27Ø CONTINUE                                                          LIN 144Ø
C SOLUTION OF LINEAR SYSTEM IS COMPLETE. RETURN FOR OPTION=1,3.       LIN 145Ø
    GO TO (28Ø, 29Ø, 28Ø, 29Ø), OPTION                                LIN 146Ø
28Ø RETURN                                                            LIN 147Ø
29Ø GO TO (13Ø, 3ØØ), ISWIT                                           LIN 148Ø
C CALCULATE ERRORS BASED ON CORRECTION.                               LIN 149Ø
3ØØ NORMX = Ø.ØDØ                                                     LIN 15ØØ
    NORME = Ø.ØDØ                                                     LIN 151Ø
    DO 31Ø I=1,N                                                      LIN 152Ø
      NORMX = DMAX1(NORMX,DABS(X(I)))                                 LIN 153Ø
      NORME = DMAX1(NORME,DABS(R(I)))                                 LIN 154Ø
31Ø CONTINUE                                                          LIN 155Ø
    ERROR = NORME/NORMX                                               LIN 156Ø
    RETURN                                                            LIN 157Ø
    END                                                               LIN 158Ø
```

ALGORITHM 504
GERK: Global Error Estimation for Ordinary Differential Equations [D]

L. F. SHAMPINE and H. A. WATTS

Sandia Laboratories

Key Words and Phrases: ordinary differential equations, initial value problems, global error estimation, Runge-Kutta-Fehlberg method, Fortran code GERK
CR Categories: 3.20, 5.17
Language: Fortran

DESCRIPTION

This algorithm is a complement to [1] where the theoretical development is described.

REFERENCES

1. SHAMPINE, L.F., AND WATTS, H.A. Global error estimation for ordinary differential equations. *ACM Trans. Math. Software 2*, 2 (June 1976), 172–186.

ALGORITHM

```
      SUBROUTINE GERK(F, NEQN, Y, T, TOUT, RELERR, ABSERR, IFLAG,       GER   10
     * GERROR, WORK, IWORK)                                             GER   20
C                                                                       GER   30
C     FEHLBERG FOURTH(FIFTH) ORDER RUNGE-KUTTA METHOD WITH              GER   40
C     GLOBAL ERROR ASSESSMENT                                           GER   50
C                                                                       GER   60
C     WRITTEN BY H.A.WATTS AND L.F.SHAMPINE                             GER   70
C                  SANDIA LABORATORIES                                  GER   80
C                                                                       GER   90
C    GERK IS DESIGNED TO SOLVE SYSTEMS OF DIFFERENTIAL EQUATIONS        GER  100
C    WHEN IT IS IMPORTANT TO HAVE A READILY AVAILABLE GLOBAL ERROR      GER  110
C    ESTIMATE.  PARALLEL INTEGRATION IS PERFORMED TO YIELD TWO          GER  120
C    SOLUTIONS ON DIFFERENT MESH SPACINGS AND GLOBAL EXTRAPOLATION      GER  130
C    IS APPLIED TO PROVIDE AN ESTIMATE OF THE GLOBAL ERROR IN THE       GER  140
C    MORE ACCURATE SOLUTION.                                           GER  150
C                                                                       GER  160
C    FOR IBM SYSTEM 360 AND 370 AND OTHER MACHINES OF SIMILAR           GER  170
C    ARITHMETIC CHARACTERISTICS, THIS CODE SHOULD BE CONVERTED TO       GER  180
C    DOUBLE PRECISION.                                                 GER  190
C                                                                       GER  200
C***********************************************************************GER  210
C ABSTRACT                                                              GER  220
C***********************************************************************GER  230
C                                                                       GER  240
C    SUBROUTINE  GERK  INTEGRATES A SYSTEM OF NEQN FIRST ORDER          GER  250
C    ORDINARY DIFFERENTIAL EQUATIONS OF THE FORM                        GER  260
C             DY(I)/DT = F(T,Y(1),Y(2),...,Y(NEQN))                     GER  270
C              WHERE THE Y(I) ARE GIVEN AT T .                          GER  280
C    TYPICALLY THE SUBROUTINE IS USED TO INTEGRATE FROM T TO TOUT       GER  290
C    BUT IT CAN BE USED AS A ONE-STEP INTEGRATOR TO ADVANCE THE         GER  300
C    SOLUTION A SINGLE STEP IN THE DIRECTION OF TOUT. ON RETURN, AN     GER  310
```

Received 15 August 1975.

Authors' address: Division 2642, Sandia Laboratories, Albuquerque, NM 87115.

```
C     ESTIMATE OF THE GLOBAL ERROR IN THE SOLUTION AT T IS PROVIDED      GER  320
C     AND THE PARAMETERS IN THE CALL LIST ARE SET FOR CONTINUING THE     GER  330
C     INTEGRATION. THE USER HAS ONLY TO CALL GERK AGAIN (AND PERHAPS     GER  340
C     DEFINE A NEW VALUE FOR TOUT). ACTUALLY, GERK  IS MERELY AN         GER  350
C     INTERFACING ROUTINE WHICH ALLOCATES VIRTUAL STORAGE IN THE         GER  360
C     ARRAYS WORK, IWORK AND CALLS SUBROUTINE GERKS FOR THE SOLUTION.    GER  370
C     GERKS IN TURN CALLS SUBROUTINE FEHL WHICH COMPUTES AN APPROX-      GER  380
C     IMATE SOLUTION OVER ONE STEP.                                      GER  390
C                                                                        GER  400
C     GERK  USES THE RUNGE-KUTTA-FEHLBERG (4,5)  METHOD DESCRIBED        GER  410
C     IN THE REFERENCE                                                   GER  420
C     E.FEHLBERG , LOW-ORDER CLASSICAL RUNGE-KUTTA FORMULAS WITH         GER  430
C                  STEPSIZE CONTROL , NASA TR R-315                      GER  440
C                                                                        GER  450
C                                                                        GER  460
C     THE PARAMETERS REPRESENT-                                          GER  470
C       F -- SUBROUTINE F(T,Y,YP) TO EVALUATE DERIVATIVES                GER  480
C            YP(I)=DY(I)/DT                                              GER  490
C       NEQN -- NUMBER OF EQUATIONS TO BE INTEGRATED                     GER  500
C       Y(*) -- SOLUTION VECTOR AT T                                     GER  510
C       T -- INDEPENDENT VARIABLE                                        GER  520
C       TOUT -- OUTPUT POINT AT WHICH SOLUTION IS DESIRED                GER  530
C       RELERR,ABSERR -- RELATIVE AND ABSOLUTE ERROR TOLERANCES FOR      GER  540
C             LOCAL ERROR TEST. AT EACH STEP THE CODE REQUIRES THAT      GER  550
C                  ABS(LOCAL ERROR) .LE. RELERR*ABS(Y) + ABSERR          GER  560
C             FOR EACH COMPONENT OF THE LOCAL ERROR AND SOLUTION         GER  570
C             VECTORS.                                                   GER  580
C       IFLAG -- INDICATOR FOR STATUS OF INTEGRATION                     GER  590
C       GERROR(*) -- VECTOR WHICH ESTIMATES THE GLOBAL ERROR AT T.       GER  600
C                    THAT IS, GERROR(I) APPROXIMATES  Y(I)-TRUE          GER  610
C                    SOLUTION(I).                                        GER  620
C       WORK(*) -- ARRAY TO HOLD INFORMATION INTERNAL TO GERK WHICH      GER  630
C             IS NECESSARY FOR SUBSEQUENT CALLS. MUST BE DIMENSIONED     GER  640
C             AT LEAST  3+8*NEQN                                         GER  650
C       IWORK(*) -- INTEGER ARRAY USED TO HOLD INFORMATION INTERNAL      GER  660
C             TO GERK WHICH IS NECESSARY FOR SUBSEQUENT CALLS. MUST      GER  670
C             BE DIMENSIONED AT LEAST  5                                 GER  680
C                                                                        GER  690
C                                                                        GER  700
C*************************************************************************GER  710
C  FIRST CALL TO GERK                                                    GER  720
C*************************************************************************GER  730
C                                                                        GER  740
C     THE USER MUST PROVIDE STORAGE IN HIS CALLING PROGRAM FOR THE       GER  750
C     ARRAYS IN THE CALL LIST  -   Y(NEQN), WORK(3+8*NEQN), IWORK(5),    GER  760
C     DECLARE F IN AN EXTERNAL STATEMENT, SUPPLY SUBROUTINE F(T,Y,YP)    GER  770
C     AND INITIALIZE THE FOLLOWING PARAMETERS-                           GER  780
C                                                                        GER  790
C       NEQN -- NUMBER OF EQUATIONS TO BE INTEGRATED.  (NEQN .GE. 1)     GER  800
C       Y(*) -- VECTOR OF INITIAL CONDITIONS                             GER  810
C       T -- STARTING POINT OF INTEGRATION , MUST BE A VARIABLE          GER  820
C       TOUT -- OUTPUT POINT AT WHICH SOLUTION IS DESIRED.               GER  830
C             T=TOUT IS ALLOWED ON THE FIRST CALL ONLY,IN WHICH CASE     GER  840
C             GERK RETURNS WITH IFLAG=2 IF CONTINUATION IS POSSIBLE.     GER  850
C       RELERR,ABSERR -- RELATIVE AND ABSOLUTE LOCAL ERROR TOLERANCES    GER  860
C             WHICH MUST BE NON-NEGATIVE BUT MAY BE CONSTANTS. WE CAN    GER  870
C             USUALLY EXPECT THE GLOBAL ERRORS TO BE SOMEWHAT SMALLER    GER  880
C             THAN THE REQUESTED LOCAL ERROR TOLERANCES. TO AVOID        GER  890
C             LIMITING PRECISION DIFFICULTIES THE CODE ALWAYS USES       GER  900
C             THE LARGER OF RELERR AND AN INTERNAL RELATIVE ERROR        GER  910
C             PARAMETER WHICH IS MACHINE DEPENDENT.                      GER  920
C       IFLAG -- +1,-1  INDICATOR TO INITIALIZE THE CODE FOR EACH NEW    GER  930
C             PROBLEM. NORMAL INPUT IS +1. THE USER SHOULD SET IFLAG=    GER  940
C             -1 ONLY WHEN ONE-STEP INTEGRATOR CONTROL IS ESSENTIAL.     GER  950
C             IN THIS CASE, GERK ATTEMPTS TO ADVANCE THE SOLUTION A      GER  960
C             SINGLE STEP IN THE DIRECTION OF TOUT EACH TIME IT IS       GER  970
C             CALLED. SINCE THIS MODE OF OPERATION RESULTS IN EXTRA      GER  980
C             COMPUTING OVERHEAD, IT SHOULD BE AVOIDED UNLESS NEEDED.    GER  990
C                                                                        GER 1000
C                                                                        GER 1010
C*************************************************************************GER 1020
C  OUTPUT FROM GERK                                                      GER 1030
C*************************************************************************GER 1040
C                                                                        GER 1050
C       Y(*) -- SOLUTION AT T                                            GER 1060
C       T -- LAST POINT REACHED IN INTEGRATION.                          GER 1070
C       IFLAG = 2 -- INTEGRATION REACHED TOUT.  INDICATES SUCCESSFUL     GER 1080
C                    RETURN AND IS THE NORMAL MODE FOR CONTINUING        GER 1090
C                    INTEGRATION.                                        GER 1100
C             =-2 -- A SINGLE SUCCESSFUL STEP IN THE DIRECTION OF        GER 1110
C                    TOUT HAS BEEN TAKEN. NORMAL MODE FOR CONTINUING     GER 1120
C                    INTEGRATION ONE STEP AT A TIME.                     GER 1130
C             = 3 -- INTEGRATION WAS NOT COMPLETED BECAUSE MORE THAN     GER 1140
C                    9000 DERIVATIVE EVALUATIONS WERE NEEDED. THIS       GER 1150
C                    IS APPROXIMATELY 500 STEPS.                         GER 1160
C             = 4 -- INTEGRATION WAS NOT COMPLETED BECAUSE SOLUTION      GER 1170
C                    VANISHED MAKING A PURE RELATIVE ERROR TEST          GER 1180
C                    IMPOSSIBLE. MUST USE NON-ZERO ABSERR TO CONTINUE.   GER 1190
C                    USING THE ONE-STEP INTEGRATION MODE FOR ONE STEP    GER 1200
C                    IS A GOOD WAY TO PROCEED.                           GER 1210
```

```
C            = 5 -- INTEGRATION WAS NOT COMPLETED BECAUSE REQUESTED     GER 1220
C                   ACCURACY COULD NOT BE ACHIEVED USING SMALLEST       GER 1230
C                   ALLOWABLE STEPSIZE. USER MUST INCREASE THE ERROR    GER 1240
C                   TOLERANCE BEFORE CONTINUED INTEGRATION CAN BE       GER 1250
C                   ATTEMPTED.                                          GER 1260
C            = 6 -- GERK IS BEING USED INEFFICIENTLY IN SOLVING         GER 1270
C                   THIS PROBLEM. TOO MUCH OUTPUT IS RESTRICTING THE    GER 1280
C                   NATURAL STEPSIZE CHOICE. USE THE ONE-STEP           GER 1290
C                   INTEGRATOR MODE.                                    GER 1300
C            = 7 -- INVALID INPUT PARAMETERS                            GER 1310
C                   THIS INDICATOR OCCURS IF ANY OF THE FOLLOWING IS    GER 1320
C                   SATISFIED - NEQN .LE. 0                             GER 1330
C                               T=TOUT  AND  IFLAG .NE. +1 OR -1        GER 1340
C                               RELERR OR ABSERR .LT. 0.                GER 1350
C                               IFLAG .EQ. 0 OR .LT. -2 OR .GT. 7       GER 1360
C      GERROR(*) -- ESTIMATE OF THE GLOBAL ERROR IN THE SOLUTION AT T   GER 1370
C      WORK(*),IWORK(*) -- INFORMATION WHICH IS USUALLY OF NO           GER 1380
C                   INTEREST TO THE USER BUT NECESSARY FOR SUBSEQUENT   GER 1390
C                   CALLS.  WORK(1),.. ,WORK(NEQN) CONTAIN THE FIRST    GER 1400
C                   DERIVATIVES OF THE SOLUTION VECTOR Y AT T.          GER 1410
C                   WORK(NEQN+1) CONTAINS THE STEPSIZE H TO BE          GER 1420
C                   ATTEMPTED ON THE NEXT STEP.  IWORK(1) CONTAINS      GER 1430
C                   THE DERIVATIVE EVALUATION COUNTER.                  GER 1440
C                                                                       GER 1450
C                                                                       GER 1460
C***********************************************************************GER 1470
C  SUBSEQUENT CALLS TO GERK                                             GER 1480
C***********************************************************************GER 1490
C                                                                       GER 1500
C    SUBROUTINE GERK RETURNS WITH ALL INFORMATION NEEDED TO CONTINUE    GER 1510
C    THE INTEGRATION. IF THE INTEGRATION REACHED TOUT, THE USER NEED    GER 1520
C    ONLY DEFINE A NEW TOUT AND CALL GERK AGAIN. IN THE ONE-STEP        GER 1530
C    INTEGRATOR MODE (IFLAG=-2) THE USER MUST KEEP IN MIND THAT EACH    GER 1540
C    STEP TAKEN IS IN THE DIRECTION OF THE CURRENT TOUT.  UPON          GER 1550
C    REACHING TOUT (INDICATED BY CHANGING IFLAG TO 2), THE USER MUST    GER 1560
C    THEN DEFINE A NEW TOUT AND RESET IFLAG TO -2 TO CONTINUE IN THE    GER 1570
C    ONE-STEP INTEGRATOR MODE.                                          GER 1580
C                                                                       GER 1590
C    IF THE INTEGRATION WAS NOT COMPLETED BUT THE USER STILL WANTS      GER 1600
C    TO CONTINUE (IFLAG=3 CASE), HE JUST CALLS GERK AGAIN.  THE         GER 1610
C    FUNCTION COUNTER IS THEN RESET TO 0 AND ANOTHER 9000 FUNCTION      GER 1620
C    EVALUATIONS ARE ALLOWED.                                           GER 1630
C                                                                       GER 1640
C    HOWEVER, IN THE CASE IFLAG=4, THE USER MUST FIRST ALTER THE        GER 1650
C    ERROR CRITERION TO USE A POSITIVE VALUE OF ABSERR BEFORE           GER 1660
C    INTEGRATION CAN PROCEED. IF HE DOES NOT,EXECUTION IS TERMINATED.   GER 1670
C                                                                       GER 1680
C    ALSO, IN THE CASE IFLAG=5, IT IS NECESSARY FOR THE USER TO         GER 1690
C    RESET IFLAG TO 2 (OR -2 WHEN THE ONE-STEP INTEGRATION MODE IS      GER 1700
C    BEING USED) AS WELL AS INCREASING EITHER ABSERR,RELERR OR BOTH     GER 1710
C    BEFORE THE INTEGRATION CAN BE CONTINUED. IF THIS IS NOT DONE,      GER 1720
C    EXECUTION WILL BE TERMINATED.  THE OCCURRENCE OF IFLAG=5           GER 1730
C    INDICATES A TROUBLE SPOT (SOLUTION IS CHANGING RAPIDLY,            GER 1740
C    SINGULARITY MAY BE PRESENT) AND IT OFTEN IS INADVISABLE TO         GER 1750
C    CONTINUE.                                                          GER 1760
C                                                                       GER 1770
C    IF IFLAG=6 IS ENCOUNTERED, THE USER SHOULD USE THE ONE-STEP        GER 1780
C    INTEGRATION MODE WITH THE STEPSIZE DETERMINED BY THE CODE. IF      GER 1790
C    THE USER INSISTS UPON CONTINUING THE INTEGRATION WITH GERK IN      GER 1800
C    THE INTERVAL MODE, HE MUST RESET IFLAG TO 2 BEFORE CALLING GERK    GER 1810
C    AGAIN.  OTHERWISE,EXECUTION WILL BE TERMINATED.                    GER 1820
C                                                                       GER 1830
C    IF IFLAG=7 IS OBTAINED, INTEGRATION CAN NOT BE CONTINUED UNLESS    GER 1840
C    THE INVALID INPUT PARAMETERS ARE CORRECTED.                        GER 1850
C                                                                       GER 1860
C    IT SHOULD BE NOTED THAT THE ARRAYS WORK,IWORK CONTAIN              GER 1870
C    INFORMATION REQUIRED FOR SUBSEQUENT INTEGRATION. ACCORDINGLY,      GER 1880
C    WORK AND IWORK SHOULD NOT BE ALTERED.                              GER 1890
C                                                                       GER 1900
C***********************************************************************GER 1910
C                                                                       GER 1920
      DIMENSION Y(NEQN), GERROR(NEQN), WORK(1), IWORK(5)                GER 1930
      EXTERNAL F                                                        GER 1940
C COMPUTE INDICES FOR THE SPLITTING OF THE WORK ARRAY                   GER 1950
      K1M = NEQN + 1                                                    GER 1960
      K1 = K1M + 1                                                      GER 1970
      K2 = K1 + NEQN                                                    GER 1980
      K3 = K2 + NEQN                                                    GER 1990
      K4 = K3 + NEQN                                                    GER 2000
      K5 = K4 + NEQN                                                    GER 2010
      K6 = K5 + NEQN                                                    GER 2020
      K7 = K6 + NEQN                                                    GER 2030
      K8 = K7 + NEQN                                                    GER 2040
C *********************************************************************GER 2050
C      THIS INTERFACING ROUTINE MERELY RELIEVES THE USER OF A LONG      GER 2060
C      CALLING LIST VIA THE SPLITTING APART OF TWO WORKING STORAGE      GER 2070
C      ARRAYS. IF THIS IS NOT COMPATIBLE WITH THE USERS COMPILER,       GER 2080
C      HE MUST USE GERKS DIRECTLY.                                      GER 2090
C *********************************************************************GER 2100
      CALL GERKS(F, NEQN, Y, T, TOUT, RELERR, ABSERR, IFLAG,            GER 2110
     * GERROR, WORK(1), WORK(K1M), WORK(K1), WORK(K2), WORK(K3),        GER 2120
```

```
     * WORK(K4), WORK(K5), WORK(K6), WORK(K7), WORK(K8),                GER 2130
     * WORK(K8+1), IWORK(1), IWORK(2), IWORK(3), IWORK(4), IWORK(5))    GER 2140
       RETURN                                                           GER 2150
       END                                                              GER 2160



       SUBROUTINE GERKS(F, NEQN, Y, T, TOUT, RELERR, ABSERR, IFLAG,     GER   10
     * GERROR, YP, H, F1, F2, F3, F4, F5, YG, YGP, SAVRE, SAVAE,        GER   20
     * NFE, KOP, INIT, JFLAG, KFLAG)                                    GER   30
C      FEHLBERG FOURTH(FIFTH) ORDER RUNGE-KUTTA METHOD WITH             GER   40
C      GLOBAL ERROR ASSESSMENT                                          GER   50
C ******************************************************************   GER   60
C      GERKS INTEGRATES A SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL   GER   70
C      EQUATIONS AS DESCRIBED IN THE COMMENTS FOR GERK.  THE ARRAYS     GER   80
C      YP,F1,F2,F3,F4,F5,YG AND YGP (OF DIMENSION AT LEAST NEQN) AND    GER   90
C      THE VARIABLES H,SAVRE,SAVAE,NFE,KOP,INIT,JFLAG,AND KFLAG ARE     GER  100
C      USED INTERNALLY BY THE CODE AND APPEAR IN THE CALL LIST TO       GER  110
C      ELIMINATE LOCAL RETENTION OF VARIABLES BETWEEN CALLS.            GER  120
C      ACCORDINGLY, THEY SHOULD NOT BE ALTERED. ITEMS OF POSSIBLE       GER  130
C      INTEREST ARE                                                     GER  140
C          YP - DERIVATIVE OF SOLUTION VECTOR AT T                      GER  150
C          H  - AN APPROPRIATE STEPSIZE TO BE USED FOR THE NEXT STEP    GER  160
C          NFE- COUNTER ON THE NUMBER OF DERIVATIVE FUNCTION            GER  170
C               EVALUATIONS.                                            GER  180
C ******************************************************************   GER  190
       LOGICAL HFAILD, OUTPUT                                           GER  200
       DIMENSION Y(NEQN), YP(NEQN), F1(NEQN), F2(NEQN), F3(NEQN),       GER  210
     * F4(NEQN), F5(NEQN), YG(NEQN), YGP(NEQN), GERROR(NEQN)            GER  220
       EXTERNAL F                                                       GER  230
C ******************************************************************   GER  240
C   THE COMPUTER UNIT ROUNDOFF ERROR U IS THE SMALLEST POSITIVE VALUE   GER  250
C   REPRESENTABLE IN THE MACHINE SUCH THAT  1.+ U .GT. 1.              GER  260
C                  VALUES TO BE USED ARE                               GER  270
C               U = 9.5E-7          FOR IBM 360/370                    GER  280
C               U = 1.5E-8          FOR UNIVAC 1108                    GER  290
C               U = 7.5E-9          FOR PDP-10                         GER  300
C               U = 7.1E-15         FOR CDC 6000  SERIES               GER  310
C               U = 2.2E-16         FOR IBM 360/370  DOUBLE PRECISION  GER  320
       DATA U /       /                                                GER  330
C ******************************************************************   GER  340
C   REMIN IS A TOLERANCE THRESHOLD WHICH IS ALSO DETERMINED BY THE     GER  350
C   INTEGRATION METHOD. IN PARTICULAR, A FIFTH ORDER METHOD WILL       GER  360
C   GENERALLY NOT BE CAPABLE OF DELIVERING ACCURACIES NEAR LIMITING    GER  370
C   PRECISION ON COMPUTERS WITH LONG WORDLENGTHS.                      GER  380
       DATA REMIN /3.E-11/                                              GER  390
C ******************************************************************   GER  400
C      THE EXPENSE IS CONTROLLED BY RESTRICTING THE NUMBER              GER  410
C      OF FUNCTION EVALUATIONS TO BE APPROXIMATELY MAXNFE.              GER  420
C      AS SET,THIS CORRESPONDS TO ABOUT 500 STEPS.                      GER  430
       DATA MAXNFE /9000/                                               GER  440
C ******************************************************************   GER  450
C      CHECK INPUT PARAMETERS                                           GER  460
       IF (NEQN.LT.1) GO TO 10                                          GER  470
       IF ((RELERR.LT.0.) .OR. (ABSERR.LT.0.)) GO TO 10                 GER  480
       MFLAG = IABS(IFLAG)                                              GER  490
       IF ((MFLAG.GE.1) .AND. (MFLAG.LE.7)) GO TO 20                    GER  500
C INVALID INPUT                                                         GER  510
   10  IFLAG = 7                                                        GER  520
       RETURN                                                           GER  530
C IS THIS THE FIRST CALL                                                GER  540
   20  IF (MFLAG.EQ.1) GO TO 70                                         GER  550
C CHECK CONTINUATION POSSIBILITIES                                      GER  560
       IF (T.EQ.TOUT) GO TO 10                                          GER  570
       IF (MFLAG.NE.2) GO TO 30                                         GER  580
C IFLAG = +2 OR -2                                                      GER  590
       IF (INIT.EQ.0) GO TO 60                                          GER  600
       IF (KFLAG.EQ.3) GO TO 50                                         GER  610
       IF ((KFLAG.EQ.4) .AND. (ABSERR.EQ.0.)) GO TO 40                  GER  620
       IF ((KFLAG.EQ.5) .AND. (RELERR.LE.SAVRE) .AND.                   GER  630
     * (ABSERR.LE.SAVAE)) GO TO 40                                      GER  640
       GO TO 70                                                         GER  650
C IFLAG = 3,4,5,6, OR 7                                                 GER  660
   30  IF (IFLAG.EQ.3) GO TO 50                                         GER  670
       IF ((IFLAG.EQ.4) .AND. (ABSERR.GT.0.)) GO TO 60                  GER  680
C INTEGRATION CANNOT BE CONTINUED SINCE USER DID NOT RESPOND TO         GER  690
C THE INSTRUCTIONS PERTAINING TO IFLAG=4,5,6 OR 7                       GER  700
   40  STOP                                                             GER  710
C ******************************************************************   GER  720
C      RESET FUNCTION EVALUATION COUNTER                                GER  730
   50  NFE = 0                                                          GER  740
       IF (MFLAG.EQ.2) GO TO 70                                         GER  750
C RESET FLAG VALUE FROM PREVIOUS CALL                                   GER  760
   60  IFLAG = JFLAG                                                    GER  770
C SAVE INPUT IFLAG AND SET CONTINUATION FLAG VALUE FOR SUBSEQUENT       GER  780
C INPUT CHECKING                                                        GER  790
   70  JFLAG = IFLAG                                                    GER  800
       KFLAG = 0                                                        GER  810
C SAVE RELERR AND ABSERR FOR CHECKING INPUT ON SUBSEQUENT CALLS         GER  820
       SAVRE = RELERR                                                   GER  830
       SAVAE = ABSERR                                                   GER  840
```

```
C RESTRICT RELATIVE ERROR TOLERANCE TO BE AT LEAST AS LARGE AS          GER  850
C 32U+REMIN TO AVOID LIMITING PRECISION DIFFICULTIES ARISING            GER  860
C FROM IMPOSSIBLE ACCURACY REQUESTS                                     GER  870
      RER = AMAX1(RELERR,32.*U+REMIN)                                   GER  880
      U26 = 26.*U                                                       GER  890
      DT = TOUT - T                                                     GER  900
      IF (MFLAG.EQ.1) GO TO 80                                          GER  910
      IF (INIT.EQ.0) GO TO 90                                           GER  920
      GO TO 110                                                         GER  930
C *********************************************************************GER  940
C      INITIALIZATION --                                                GER  950
C                       SET INITIALIZATION COMPLETION INDICATOR,INIT    GER  960
C                       SET INDICATOR FOR TOO MANY OUTPUT POINTS,KOP    GER  970
C                       EVALUATE INITIAL DERIVATIVES                    GER  980
C                       COPY INITIAL VALUES AND DERIVATIVES FOR THE     GER  990
C                             PARALLEL SOLUTION                         GER 1000
C                       SET COUNTER FOR FUNCTION EVALUATIONS,NFE        GER 1010
C                       ESTIMATE STARTING STEPSIZE                      GER 1020
   80 INIT = 0                                                          GER 1030
      KOP = 0                                                           GER 1040
      A = T                                                             GER 1050
      CALL F(A, Y, YP)                                                  GER 1060
      NFE = 1                                                           GER 1070
      IF (T.NE.TOUT) GO TO 90                                           GER 1080
      IFLAG = 2                                                         GER 1090
      RETURN                                                            GER 1100
   90 INIT = 1                                                          GER 1110
      H = ABS(DT)                                                       GER 1120
      TOLN = 0.                                                         GER 1130
      DO 100 K=1,NEQN                                                   GER 1140
        YG(K) = Y(K)                                                    GER 1150
        YGP(K) = YP(K)                                                  GER 1160
        TOL = RER*ABS(Y(K)) + ABSERR                                    GER 1170
        IF (TOL.LE.0.) GO TO 100                                        GER 1180
        TOLN = TOL                                                      GER 1190
        YPK = ABS(YP(K))                                                GER 1200
        IF (YPK*H**5.GT.TOL) H = (TOL/YPK)**0.2                         GER 1210
  100 CONTINUE                                                          GER 1220
      IF (TOLN.LE.0.) H = 0.                                            GER 1230
      H = AMAX1(H,U26*AMAX1(ABS(T),ABS(DT)))                            GER 1240
C *********************************************************************GER 1250
C      SET STEPSIZE FOR INTEGRATION IN THE DIRECTION FROM T TO TOUT     GER 1260
  110 H = SIGN(H,DT)                                                    GER 1270
C TEST TO SEE IF GERK IS BEING SEVERELY IMPACTED BY TOO MANY            GER 1280
C OUTPUT POINTS                                                         GER 1290
      IF (ABS(H).GT.2.*ABS(DT)) KOP = KOP + 1                           GER 1300
      IF (KOP.NE.100) GO TO 120                                         GER 1310
      KOP = 0                                                           GER 1320
      IFLAG = 6                                                         GER 1330
      RETURN                                                            GER 1340
  120 IF (ABS(DT).GT.U26*ABS(T)) GO TO 140                              GER 1350
C IF TOO CLOSE TO OUTPUT POINT,EXTRAPOLATE AND RETURN                   GER 1360
      DO 130 K=1,NEQN                                                   GER 1370
        YG(K) = YG(K) + DT*YGP(K)                                       GER 1380
        Y(K) = Y(K) + DT*YP(K)                                          GER 1390
  130 CONTINUE                                                          GER 1400
      A = TOUT                                                          GER 1410
      CALL F(A, YG, YGP)                                                GER 1420
      CALL F(A, Y, YP)                                                  GER 1430
      NFE = NFE + 2                                                     GER 1440
      GO TO 230                                                         GER 1450
C INITIALIZE OUTPUT POINT INDICATOR                                     GER 1460
  140 OUTPUT = .FALSE.                                                  GER 1470
C TO AVOID PREMATURE UNDERFLOW IN THE ERROR TOLERANCE FUNCTION,         GER 1480
C SCALE THE ERROR TOLERANCES                                            GER 1490
      SCALE = 2./RER                                                    GER 1500
      AE = SCALE*ABSERR                                                 GER 1510
C *********************************************************************GER 1520
C *********************************************************************GER 1530
C      STEP BY STEP INTEGRATION                                         GER 1540
  150 HFAILD = .FALSE.                                                  GER 1550
C SET SMALLEST ALLOWABLE STEPSIZE                                       GER 1560
      HMIN = U26*ABS(T)                                                 GER 1570
C ADJUST STEPSIZE IF NECESSARY TO HIT THE OUTPUT POINT.                 GER 1580
C LOOK AHEAD TWO STEPS TO AVOID DRASTIC CHANGES IN THE STEPSIZE         GER 1590
C AND THUS LESSEN THE IMPACT OF OUTPUT POINTS ON THE CODE.              GER 1600
      DT = TOUT - T                                                     GER 1610
      IF (ABS(DT).GE.2.*ABS(H)) GO TO 170                               GER 1620
      IF (ABS(DT).GT.ABS(H)) GO TO 160                                  GER 1630
C THE NEXT SUCCESSFUL STEP WILL COMPLETE THE INTEGRATION TO THE         GER 1640
C OUTPUT POINT                                                          GER 1650
      OUTPUT = .TRUE.                                                   GER 1660
      H = DT                                                            GER 1670
      GO TO 170                                                         GER 1680
  160 H = 0.5*DT                                                        GER 1690
C *********************************************************************GER 1700
C      CORE INTEGRATOR FOR TAKING A SINGLE STEP                         GER 1710
C *********************************************************************GER 1720
C      THE TOLERANCES HAVE BEEN SCALED TO AVOID PREMATURE UNDERFLOW     GER 1730
C      IN COMPUTING THE ERROR TOLERANCE FUNCTION ET.                    GER 1740
C      TO AVOID PROBLEMS WITH ZERO CROSSINGS, RELATIVE ERROR IS         GER 1750
```

```
C       MEASURED USING THE AVERAGE OF THE MAGNITUDES OF THE SOLUTION     GER 1760
C       AT THE BEGINNING AND END OF A STEP.                              GER 1770
C       THE ERROR ESTIMATE FORMULA HAS BEEN GROUPED TO CONTROL LOSS OF   GER 1780
C       SIGNIFICANCE.                                                    GER 1790
C       TO DISTINGUISH THE VARIOUS ARGUMENTS, H IS NOT PERMITTED         GER 1800
C       TO BECOME SMALLER THAN 26 UNITS OF ROUNDOFF IN T.                GER 1810
C       PRACTICAL LIMITS ON THE CHANGE IN THE STEPSIZE ARE ENFORCED TO   GER 1820
C       SMOOTH THE STEPSIZE SELECTION PROCESS AND TO AVOID EXCESSIVE     GER 1830
C       CHATTERING ON PROBLEMS HAVING DISCONTINUITIES.                   GER 1840
C       TO PREVENT UNNECESSARY FAILURES, THE CODE USES 9/10 THE          GER 1850
C       STEPSIZE IT ESTIMATES WILL SUCCEED.                              GER 1860
C       AFTER A STEP FAILURE, THE STEPSIZE IS NOT ALLOWED TO INCREASE    GER 1870
C       FOR THE NEXT ATTEMPTED STEP.  THIS MAKES THE CODE MORE           GER 1880
C       EFFICIENT ON PROBLEMS HAVING DISCONTINUITIES AND MORE            GER 1890
C       EFFECTIVE IN GENERAL SINCE LOCAL EXTRAPOLATION IS BEING USED     GER 1900
C       AND THE ERROR ESTIMATE MAY BE UNRELIABLE OR UNACCEPTABLE WHEN    GER 1910
C       A STEP FAILS.                                                    GER 1920
C ***************************************************************      GER 1930
C       TEST NUMBER OF DERIVATIVE FUNCTION EVALUATIONS.                  GER 1940
C       IF OKAY,TRY TO ADVANCE THE INTEGRATION FROM T TO T+H             GER 1950
  170 IF (NFE.LE.MAXNFE) GO TO 180                                       GER 1960
C TOO MUCH WORK                                                          GER 1970
      IFLAG = 3                                                          GER 1980
      KFLAG = 3                                                          GER 1990
      RETURN                                                             GER 2000
C ADVANCE AN APPROXIMATE SOLUTION OVER ONE STEP OF LENGTH H              GER 2010
  180 CALL FEHL(F, NEQN, YG, T, H, YGP, F1, F2, F3, F4, F5, F1)          GER 2020
      NFE = NFE + 5                                                      GER 2030
C COMPUTE AND TEST ALLOWABLE TOLERANCES VERSUS LOCAL ERROR               GER 2040
C ESTIMATES AND REMOVE SCALING OF TOLERANCES. NOTE THAT RELATIVE         GER 2050
C ERROR IS MEASURED WITH RESPECT TO THE AVERAGE MAGNITUDES OF THE        GER 2060
C OF THE SOLUTION AT THE BEGINNING AND END OF THE STEP.                  GER 2070
      EEOET = 0.                                                         GER 2080
      DO 200 K=1,NEQN                                                    GER 2090
        ET = ABS(YG(K)) + ABS(F1(K)) + AE                                GER 2100
        IF (ET.GT.0.) GO TO 190                                          GER 2110
C INAPPROPRIATE ERROR TOLERANCE                                          GER 2120
        IFLAG = 4                                                        GER 2130
        KFLAG = 4                                                        GER 2140
        RETURN                                                           GER 2150
  190   EE = ABS((-2090.*YGP(K)+(21970.*F3(K)-15048.*F4(K)))             GER 2160
     *   +(22528.*F2(K)-27360.*F5(K)))                                   GER 2170
        EEOET = AMAX1(EEOET,EE/ET)                                       GER 2180
  200 CONTINUE                                                           GER 2190
      ESTTOL = ABS(H)*EEOET*SCALE/752400.                                GER 2200
      IF (ESTTOL.LE.1.) GO TO 210                                        GER 2210
C UNSUCCESSFUL STEP                                                      GER 2220
C                   REDUCE THE STEPSIZE , TRY AGAIN                      GER 2230
C                   THE DECREASE IS LIMITED TO A FACTOR OF 1/10          GER 2240
      HFAILD = .TRUE.                                                    GER 2250
      OUTPUT = .FALSE.                                                   GER 2260
      S = 0.1                                                            GER 2270
      IF (ESTTOL.LT.59049.) S = 0.9/ESTTOL**0.2                          GER 2280
      H = S*H                                                            GER 2290
      IF (ABS(H).GT.HMIN) GO TO 170                                      GER 2300
C REQUESTED ERROR UNATTAINABLE AT SMALLEST ALLOWABLE STEPSIZE            GER 2310
      IFLAG = 5                                                          GER 2320
      KFLAG = 5                                                          GER 2330
      RETURN                                                             GER 2340
C SUCCESSFUL STEP                                                        GER 2350
C                    STORE ONE-STEP SOLUTION YG AT T+H                   GER 2360
C                    AND EVALUATE DERIVATIVES THERE                      GER 2370
  210 TS = T                                                             GER 2380
      T = T + H                                                          GER 2390
      DO 220 K=1,NEQN                                                    GER 2400
        YG(K) = F1(K)                                                    GER 2410
  220 CONTINUE                                                           GER 2420
      A = T                                                              GER 2430
      CALL F(A, YG, YGP)                                                 GER 2440
      NFE = NFE + 1                                                      GER 2450
C NOW ADVANCE THE Y SOLUTION OVER TWO STEPS OF                           GER 2460
C LENGTH H/2 AND EVALUATE DERIVATIVES THERE                              GER 2470
      HH = 0.5*H                                                         GER 2480
      CALL FEHL(F, NEQN, Y, TS, HH, YP, F1, F2, F3, F4, F5, Y)           GER 2490
      TS = TS + HH                                                       GER 2500
      A = TS                                                             GER 2510
      CALL F(A, Y, YP)                                                   GER 2520
      CALL FEHL(F, NEQN, Y, TS, HH, YP, F1, F2, F3, F4, F5, Y)           GER 2530
      A = T                                                              GER 2540
      CALL F(A, Y, YP)                                                   GER 2550
      NFE = NFE + 12                                                     GER 2560
C CHOOSE NEXT STEPSIZE                                                   GER 2570
C THE INCREASE IS LIMITED TO A FACTOR OF 5                               GER 2580
C IF STEP FAILURE HAS JUST OCCURRED, NEXT                                GER 2590
C    STEPSIZE IS NOT ALLOWED TO INCREASE                                 GER 2600
      S = 5.                                                             GER 2610
      IF (ESTTOL.GT.1.889568E-4) S = 0.9/ESTTOL**0.2                     GER 2620
      IF (HFAILD) S = AMIN1(S,1.)                                        GER 2630
      H = SIGN(AMAX1(S*ABS(H),HMIN),H)                                   GER 2640
C ***************************************************************      GER 2650
C       END OF CORE INTEGRATOR                                           GER 2660
```

```
C ***********************************************************************  GER 2670
C      SHOULD WE TAKE ANOTHER STEP                                           GER 2680
      IF (OUTPUT) GO TO 230                                                  GER 2690
      IF (IFLAG.GT.0) GO TO 150                                              GER 2700
C ***********************************************************************  GER 2710
C ***********************************************************************  GER 2720
C      INTEGRATION SUCCESSFULLY COMPLETED                                    GER 2730
C      ONE-STEP MODE                                                         GER 2740
      IFLAG = -2                                                             GER 2750
      GO TO 240                                                              GER 2760
C INTERVAL MODE                                                              GER 2770
  230 T = TOUT                                                               GER 2780
      IFLAG = 2                                                              GER 2790
  240 DO 250 K=1,NEQN                                                        GER 2800
        GERROR(K) = (YG(K)-Y(K))/31.                                         GER 2810
  250 CONTINUE                                                               GER 2820
      RETURN                                                                 GER 2830
      END                                                                    GER 2840


      SUBROUTINE FEHL(F, NEQN, Y, T, H, YP, F1, F2, F3, F4, F5, S)           FEH   10
C      FEHLBERG FOURTH-FIFTH ORDER RUNGE-KUTTA METHOD                        FEH   20
C ***********************************************************************  FEH   30
C     FEHL INTEGRATES A SYSTEM OF NEQN FIRST ORDER                           FEH   40
C     ORDINARY DIFFERENTIAL EQUATIONS OF THE FORM                            FEH   50
C              DY(I)/DT=F(T,Y(1),---,Y(NEQN))                                FEH   60
C     WHERE THE INITIAL VALUES Y(I) AND THE INITIAL DERIVATIVES              FEH   70
C     YP(I) ARE SPECIFIED AT THE STARTING POINT T. FEHL ADVANCES             FEH   80
C     THE SOLUTION OVER THE FIXED STEP H AND RETURNS                         FEH   90
C     THE FIFTH ORDER (SIXTH ORDER ACCURATE LOCALLY) SOLUTION                FEH  100
C     APPROXIMATION AT T+H IN ARRAY S(I).                                    FEH  110
C     F1,---,F5 ARE ARRAYS OF DIMENSION NEQN WHICH ARE NEEDED                FEH  120
C     FOR INTERNAL STORAGE.                                                  FEH  130
C     THE FORMULAS HAVE BEEN GROUPED TO CONTROL LOSS OF SIGNIFICANCE.        FEH  140
C     FEHL SHOULD BE CALLED WITH AN H NOT SMALLER THAN 13 UNITS OF           FEH  150
C     ROUNDOFF IN T SO THAT THE VARIOUS INDEPENDENT ARGUMENTS CAN BE         FEH  160
C     DISTINGUISHED.                                                         FEH  170
C ***********************************************************************  FEH  180
      DIMENSION Y(NEQN), YP(NEQN), F1(NEQN), F2(NEQN), F3(NEQN),             FEH  190
     * F4(NEQN), F5(NEQN), S(NEQN)                                           FEH  200
      CH = 0.25*H                                                            FEH  210
      DO 10 K=1,NEQN                                                         FEH  220
        F5(K) = Y(K) + CH*YP(K)                                              FEH  230
   10 CONTINUE                                                               FEH  240
      CALL F(T+0.25*H, F5, F1)                                               FEH  250
      CH = 0.09375*H                                                         FEH  260
      DO 20 K=1,NEQN                                                         FEH  270
        F5(K) = Y(K) + CH*(YP(K)+3.*F1(K))                                   FEH  280
   20 CONTINUE                                                               FEH  290
      CALL F(T+0.375*H, F5, F2)                                              FEH  300
      CH = H/2197.                                                           FEH  310
      DO 30 K=1,NEQN                                                         FEH  320
        F5(K) = Y(K) + CH*(1932.*YP(K)+(7296.*F2(K)-7200.*F1(K)))            FEH  330
   30 CONTINUE                                                               FEH  340
      CALL F(T+12./13.*H, F5, F3)                                            FEH  350
      CH = H/4104.                                                           FEH  360
      DO 40 K=1,NEQN                                                         FEH  370
        F5(K) = Y(K) + CH*((8341.*YP(K)-845.*F3(K))+(29440.*F2(K)            FEH  380
     *   -32832.*F1(K)))                                                     FEH  390
   40 CONTINUE                                                               FEH  400
      CALL F(T+H, F5, F4)                                                    FEH  410
      CH = H/20520.                                                          FEH  420
      DO 50 K=1,NEQN                                                         FEH  430
        F1(K) = Y(K) + CH*((-6080.*YP(K)+(9295.*F3(K)-5643.*F4(K)))          FEH  440
     *   +(41040.*F1(K)-28352.*F2(K)))                                       FEH  450
   50 CONTINUE                                                               FEH  460
      CALL F(T+0.5*H, F1, F5)                                                FEH  470
C COMPUTE APPROXIMATE SOLUTION AT T+H                                        FEH  480
      CH = H/7618050.                                                        FEH  490
      DO 60 K=1,NEQN                                                         FEH  500
        S(K) = Y(K) + CH*((902880.*YP(K)+(3855735.*F3(K)-1371249.*           FEH  510
     *   F4(K)))+(3953664.*F2(K)+277020.*F5(K)))                             FEH  520
   60 CONTINUE                                                               FEH  530
      RETURN                                                                 FEH  540
      END                                                                    FEH  550
```

ALGORITHM 505
A List Insertion Sort for Keys With Arbitrary Key Distribution [S20]

WOLFGANG JANKO
Hochschule für Welthandel, Austria

Key Words and Phrases: sorting, searching, linked lists, lists, insertion
CR Categories: 5.31, 4.49, 3.74
Language: Basic Fortran

DESCRIPTION

The description of this algorithm, the experimental results, and the references are given in the author's paper, "A List Insertion Sort for Keys With Arbitrary Key Distribution," *ACM Trans. Math. Software 2*, 2(June 1976), 143–153.

ALGORITHM

```
      SUBROUTINE SPN(K, L, II, JJ, MIN)                                  SPN   10
C          **SAMPLE SEARCH SORT**                                         SPN   20
C A LIST INSERTION SORT TO BUILD UP A SINGLE CIRCULARLY                   SPN   30
C LINKED LIST.                                                            SPN   40
C THIS ALGORITHM SORTS THE KEYS K(II),K(II+1),...,K(JJ) IN                SPN   50
C ASCENDING SORTING ORDER BY THE MEANS OF THE POINTER FIELDS              SPN   60
C L(II),L(II+1),....,L(JJ).                                               SPN   70
C THE EXPECTED NUMBER OF NECESSARY COMPARISIONS AND                       SPN   80
C ACCESSES IS OF ORDER N**1.5 (N=JJ-II+1).                                SPN   90
C WITH SINGLE KEYS THE ALGORITHM IS PROPORTIONAL IN SORTING               SPN  100
C TIME TO A SHELL SORT ALGORITHM.                                         SPN  110
C ISTRT  AUXILIARY VARIABLE IN RANDOM SEARCH                              SPN  120
C K      ARRAY OF N KEYS.                                                 SPN  130
C L      ARRAY FOR N POINTERS.                                            SPN  140
C IPOI   VARIABLE WHICH IS USED TO MANIPULATE POINTERS.                   SPN  150
C KDIFF  DISTANCE OF THE KEY FOUND IN RANDOM SEARCH                       SPN  160
C            AND THE KEY WHICH SHALL BE INSERTED.                         SPN  170
C KEY,MADIN  ADDRESSES USED IN SEQUENTIAL AND RANDOM SEARCH.              SPN  180
C MAX,MIN    VARIABLES TO STORE THE ADDRESS OF THE                        SPN  190
C            KEYS WITH THE SMALLEST AND LARGEST VALUE.                    SPN  200
C N      NUMBER OF KEYS.                                                  SPN  210
C IRT    STEP WIDTH IN RANDOM SEARCH.                                     SPN  220
C TAB    ARRAY OF VALUES WHICH DETERMINES THE SAMPLE SIZE.                SPN  230
C THE ALGORITHM IS STABLE.                                                SPN  240
C MIN RETURNS THE ADDRESS OF THE FIRST RECORD IN SORTING                  SPN  250
C ORDER OF THE LIST.                                                      SPN  260
      INTEGER TAB(57)                                                     SPN  270
      DIMENSION K(1), L(1)                                                SPN  280
      DATA TAB(1), TAB(2), TAB(3), TAB(4), TAB(5), TAB(6), TAB(7),        SPN  290
     * TAB(8), TAB(9), TAB(10), TAB(11), TAB(12), TAB(13), TAB(14),        SPN  300
     * TAB(15), TAB(16), TAB(17), TAB(18), TAB(19), TAB(20),               SPN  310
     * TAB(21), TAB(22), TAB(23), TAB(24), TAB(25), TAB(26),               SPN  320
     * TAB(27), TAB(28), TAB(29), TAB(30), TAB(31), TAB(32),               SPN  330
     * TAB(33), TAB(34), TAB(35), TAB(36), TAB(37), TAB(38),               SPN  340
     * TAB(39), TAB(40), TAB(41), TAB(42), TAB(43), TAB(44),               SPN  350
     * TAB(45), TAB(46), TAB(47), TAB(48), TAB(49), TAB(50),               SPN  360
```

Received 2 July 1974.

Author's address: Institut für Statistik und Mathematik, Hochschule für Welthandel, Franz Klein-Gasse 1, A-1190, Wien, Austria.

```
     * TAB(51), TAB(52), TAB(53), TAB(54), TAB(55), TAB(56),          SPN  370
     * TAB(57) /3,5,10,17,26,37,50,65,82,101,122,145,170,197,226,     SPN  380
     * 257,290,325,362,401,442,485,530,577,626,677,730,785,842,901,   SPN  390
     * 962,1025,1090,1157,1226,1297,1370,1445,1522,1601,1682,1765,    SPN  400
     * 1850,1937,2026,2117,2210,2305,2402,2501,2602,2705,2810,2917,   SPN  410
     * 3026,3137,3250/                                                SPN  420
C  INITIALISATION (PART I)                                           SPN  430
C -------------------------------------------------------------      SPN  440
      MIN = II                                                       SPN  450
      MAX = II                                                       SPN  460
      L(II) = II                                                     SPN  470
      IF (JJ-II) 170, 170, 10                                        SPN  480
   10 KMIN = K(MIN)                                                  SPN  490
      KMAX = KMIN                                                    SPN  500
      IA = II + 1                                                    SPN  510
      IRT = 1                                                        SPN  520
      ITAB = TAB(1) + II - 1                                         SPN  530
      ISTRT = II                                                     SPN  540
C -------------------------------------------------------------      SPN  550
C  INSERTION LOOP (PART II)                                         SPN  560
      DO 160 J=IA,JJ                                                 SPN  570
C CORRECTION OF THE SAMPLE SIZE (IF NECESSARY)                       SPN  580
      IF (J-ITAB) 30, 20, 20                                         SPN  590
C -------------------------------------------------------------      SPN  600
   20 IRT = IRT + 1                                                  SPN  610
      ISTRT = ISTRT + 1                                              SPN  620
      ITAB = TAB(IRT) + II - 1                                       SPN  630
C -------------------------------------------------------------      SPN  640
   30 KJ = K(J)                                                      SPN  650
C PROVISION FOR ARISING LARGEST AND SMALLEST KEYS (1).               SPN  660
      IF (KJ-KMAX) 40, 60, 60                                        SPN  670
C -------------------------------------------------------------      SPN  680
   40 IF (KJ-KMIN) 70, 50, 90                                        SPN  690
C -------------------------------------------------------------      SPN  700
C  KEY WHICH SHALL BE INSERTED IS EQUAL TO THE MINIMAL               SPN  710
C  KEY SORTED SO FAR (2).                                            SPN  720
C -------------------------------------------------------------      SPN  730
   50 MADIN = MIN                                                    SPN  740
      MIN = J                                                        SPN  750
      GO TO 130                                                      SPN  760
C -------------------------------------------------------------      SPN  770
C  KEY WHICH SHALL BE INSERTED IS EQUAL TO OR LARGER THAN THE        SPN  780
C  MAXIMAL KEY SORTED SO FAR (3).                                    SPN  790
C -------------------------------------------------------------      SPN  800
   60 I = MAX                                                        SPN  810
      MAX = J                                                        SPN  820
      KMAX = K(J)                                                    SPN  830
      GO TO 80                                                       SPN  840
C -------------------------------------------------------------      SPN  850
C  KEY FOR INSERTION IS SMALLER THAN THE SMALLEST KEY SORTED         SPN  860
C  SO FAR (4).                                                       SPN  870
C -------------------------------------------------------------      SPN  880
   70 I = MAX                                                        SPN  890
      MIN = J                                                        SPN  900
      KMIN = K(J)                                                    SPN  910
C -------------------------------------------------------------      SPN  920
C  INSERTION OF THE RECORD AND CORRECTION OF THE VALUE WHICH         SPN  930
C  DETERMINES THE SAMPLE SIZE (5).                                   SPN  940
C -------------------------------------------------------------      SPN  950
   80 IPOI = L(I)                                                    SPN  960
      L(I) = J                                                       SPN  970
      L(J) = IPOI                                                    SPN  980
      GO TO 160                                                      SPN  990
C -------------------------------------------------------------      SPN 1000
C  INITIALISATION FOR RANDOM SEARCH (FIRST STEP) (6).                SPN 1010
C -------------------------------------------------------------      SPN 1020
   90 MADIN = MIN                                                    SPN 1030
      IPOI = J - 1                                                   SPN 1040
      I = KJ - KMIN                                                  SPN 1050
C -------------------------------------------------------------      SPN 1060
C  RANDOM SEARCH (7).                                                SPN 1070
C  PART 1 (7A).                                                      SPN 1080
C -------------------------------------------------------------      SPN 1090
      DO 120 KEY=ISTRT,IPOI,IRT                                      SPN 1100
        KDIFF = KJ - K(KEY)                                          SPN 1110
        IF (KDIFF) 120, 140, 100                                     SPN 1120
C -------------------------------------------------------------      SPN 1130
  100   IF (I-KDIFF) 120, 120, 110                                   SPN 1140
C -------------------------------------------------------------      SPN 1150
C  PART 2.. STORE NEAREST KEY FOUND SO FAR (7B)                      SPN 1160
C -------------------------------------------------------------      SPN 1170
  110   I = KDIFF                                                    SPN 1180
        MADIN = KEY                                                  SPN 1190
C -------------------------------------------------------------      SPN 1200
  120   CONTINUE                                                     SPN 1210
C -------------------------------------------------------------      SPN 1220
C  SEQUENTIAL SEARCH (8).                                            SPN 1230
C -------------------------------------------------------------      SPN 1240
  130   IPOI = MADIN                                                 SPN 1250
        MADIN = L(IPOI)                                              SPN 1260
        IF (KJ-K(MADIN)) 150, 130, 130                               SPN 1270
```

```
C ------------------------------------------------------------          SPN 1280
C  KEY SEARCHED FOR WAS FOUND IN RANDOM SEARCH                         SPN 1290
C ------------------------------------------------------------          SPN 1300
  140   MADIN = KEY                                                    SPN 1310
        GO TO 130                                                      SPN 1320
C ------------------------------------------------------------          SPN 1330
C  INSERT (9)                                                          SPN 1340
C ------------------------------------------------------------          SPN 1350
  150   L(IPOI) = J                                                    SPN 1360
        L(J) = MADIN                                                   SPN 1370
C ------------------------------------------------------------          SPN 1380
  160 CONTINUE                                                         SPN 1390
C ------------------------------------------------------------          SPN 1400
  170 MIN = L(MAX)                                                     SPN 1410
      L(MAX) = 0                                                       SPN 1420
      RETURN                                                           SPN 1430
      END                                                              SPN 1440
```

ALGORITHM 506
HQR3 and EXCHNG: Fortran Subroutines for Calculating and Ordering the Eigenvalues of a Real Upper Hessenberg Matrix [F2]

G. W. STEWART
University of Maryland

Key Words and Phrases: eigenvalue, QR-algorithm
CR Categories: 5.14
Language: Fortran

DESCRIPTION

1. Usage

HQR3 is a Fortran subroutine to reduce a real upper Hessenberg matrix A to quasi-triangular form B by a unitary similarity transformation U:

$$B = U^T A U.$$

The diagonal of B consists of 1×1 and 2×2 blocks as illustrated below:

$$\begin{bmatrix} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x & x \end{bmatrix}$$

The 1×1 blocks contain the real eigenvalues of A, and the 2×2 blocks contain the complex eigenvalues, a conjugate pair to each block. The blocks are ordered so that the eigenvalues appear in descending order of absolute value along the diagonal. The transformation U is postmultiplied into an array V, which presumably contains earlier transformations performed on A.

The decomposition produced by *HQR3* differs from the one produced by the *EISPACK* subroutine *HQR2* [2] in that the eigenvalues of the final quasi-triangular matrix are ordered. This ordering makes the decomposition essentially unique, which is important in some applications (e.g. see [4]). It should also be noted that when the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ of A are ordered so that $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$, and if $|\lambda_i| > |\lambda_{i+1}|$, then the first i columns of U form an orthonormal basis for the invariant subspace corresponding to $\lambda_1, \lambda_2, \ldots, \lambda_i$. In applications where it is desired to work with the matrix A in such a dominant invariant subspace,

Received 22 August 1975 and 14 October 1975.

This work was supported in part by the Office of Naval Research under Contract N0014-67-A-0218-0018.
Author's address: Department of Computer Science, University of Maryland, College Park, MD 20742.

*HQR*3 provides a convenient means for calculating a basis. The corresponding leading principal submatrix of B is a representation of A in that subspace. When an ordered quasi-triangular form is not required, the *EISPACK* program *HQR*2 will be slightly more efficient than *HQR*3.

The calling sequence for *HQR*3 is:

```
CALL HQR3 (A,V,N,NLOW,NUP,EPS,ER,EI,TYPE,NA,NV)
```

with (parameters preceded by an asterisk are altered by the subroutine):

*A	A doubly subscripted real array containing the matrix to be reduced. On return, A contains the final quasi-triangular matrix. The elements of the array below the third subdiagonal are unaltered by the subroutine.
*V	A doubly subscripted real array into which the reducing transformation is postmultiplied.
N	An integer containing the order of A and V.
NLOW, NUP	Integers prescribing what part of A is to be reduced. Specifically A(NLOW,NLOW−1) and A(NUP+1,NUP) are assumed zero and only the block from NLOW through NUP is reduced. However, the transformation is performed on all of the matrix A so that the final result is similar to A.
EPS	Convergence criterion. Maximal accuracy will be attained if EPS is set to β^{-t}, where β is the base of the floating-point word and t is the length of its fraction. Smaller values of EPS will increase the amount of work without significantly improving the accuracy.
*ER	A singly subscripted real array containing the real parts of the eigenvalues.
*EI	A singly subscripted real array containing the imaginary parts of the eigenvalues.
*TYPE	A singly subscripted integer array whose ith entry is: 0 if the ith eigenvalue is real; 1 if the ith eigenvalue is complex with positive imaginary part; 2 if the ith eigenvalue is complex with negative imaginary part; −1 if the ith eigenvalue was not successfully calculated. The entry 1 is always followed by a 2. Only elements NLOW through NUP of ER, EI, and TYPE are set by *HQR*3.
NA	The first dimension of the array A.
NV	The first dimension of the array V.

*HQR*3 can be used together with the *EISPACK* programs *ORTHES* and *ORTRAN* [2] to reduce a full matrix A to quasi-triangular form with the eigenvalues appearing in descending order of absolute value along the diagnonal,

```
CALL ORTHES(NA,N,NLOW,NUP,A,P)
CALL ORTRAN(NA,N,NLOW,NUP,A,P,V)
CALL HQR3(A,V,N,NLOW,NUP,EPS,ER,EI,TYPE,NA,NA)
```

where P is a singly subscripted scratch array of order N.

*HQR*3 requires the subroutines *EXCHNG*, *SPLIT*, and *QRSTEP*.

EXCHNG is a Fortran subroutine to interchange consecutive 1×1 and 2×2 blocks of an upper Hessenberg matrix. Specifically it is supposed that the upper Hessenberg matrix A has a block of order $b1$ starting at the lth diagonal element and a block of order $b2$ starting at the $(l + b1)$-th diagonal element (illustrated below for $n = 5, l = 2, b1 = 2, b2 = 1$):

$$\begin{array}{cc} & \begin{matrix} l & & & & \end{matrix} \\ \begin{matrix} \\ l \\ \\ \\ \\ \end{matrix} & \begin{bmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x \end{bmatrix} \end{array}$$

EXCHNG produces an orthogonal similarity transformation W such that W^TAW has consecutive blocks of order $b2$ and $b1$ starting at the lth diagonal element (illustrated from the example above):

$$\begin{array}{cc} & \begin{matrix} l & & & & \end{matrix} \\ \begin{matrix} \\ l \\ \\ \\ \\ \end{matrix} & \begin{bmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x \end{bmatrix} \end{array}$$

The eigenvalues associated with each block are interchanged along with the blocks. The transformation W is postmultiplied into the matrix V.

EXCHNG can be used to rearrange the blocks of the quasi-triangular matrix produced by *HQR*3. For example, one might wish to cluster a group of nearly equal eigenvalues at the top of the matrix before applying a deflation technique to uncouple them from the rest of the problem.

The calling sequence for *EXCHNG* is:

```
CALL EXCHNG(A,V,N,L,B1,B2,EPS,FAIL,NB,NV)
```

with (parameters preceded by an asterisk are altered by the subroutine):

*A	A doubly subscripted real array containing the matrix whose blocks are to be interchanged. Only rows and columns L through L + B1 + B2 − 1 are transformed. Elements of the array A below the third subdiagonal are not altered.
*V	A doubly subscripted array containing the matrix V into which the reducing transformation is to be accumulated. Only columns L through L + B1 + B2 − 1 are altered.
N	An integer containing the order of A and V.
L	An integer containing the leading diagonal position of the blocks.
B1	An integer containing the size of the first block.
B2	An integer containing the size of the second block.
EPS	A convergence criterion (cf. EPS in the calling sequence for *HQR*3).
*FAIL	A logical variable that on normal return is false. If the iteration to interchange the blocks failed, it is set to true.
NA	The first dimension of the array A.
NV	The first dimension of the array V.

By repeated applications of *EXCHNG* the eigenvalues of a quasi-triangular matrix can be arranged in any desired order.

EXCHNG requires the subroutine *QRSTEP*.

2. Method and Programming Details

*HQR*3 uses the implicit double-shift *QR* algorithm to reduce A to quasi-triangular form (for the theoretical background see [3, 5]). The program is essentially a Fortran variant of part of the Algol program *hqr*2 in the *Numerische Mathematik* handbook series [1] with these differences:

(1) The calling sequence is somewhat different.

(2) Eigenvectors are not computed.

(3) The parameters NLOW and NUP are not comparable to the parameters *low* and *upp* of *hqr*2. Specifically, in *HQR*3 rows 1 through N of V are transformed; whereas in *hqr*2 only rows *low* through *upp* are transformed.

(4) The code that performs the *QR* iterations in *hqr*2 is replaced by a call to the subroutine *QRSTEP*.

(5) After a 1×1 or 2×2 block has been isolated, the subroutine *EXCHNG* is used to position it correctly among the previously isolated blocks. It should be realized that *EXCHNG* is a numerical algorithm and may change an ill-conditioned eigenvalue in a block significantly. This means that after interchanging two blocks with ill-conditioned eigenvalues that are very nearly equal in absolute value, the eigenvalues may still not be in descending order of absolute value. Since numerically the absolute values of these eigenvalues cannot be told apart, this phenomenon is not of much importance.

The convergence criterion is the same for both programs. A subdiagonal element A(I + 1, I) is regarded as negligible it it is less than or equal to EPS * (ABS(A(I, I)) + ABS(A(I + 1, I + 1))). If 10 or 20 iterations are performed without convergence, an ad hoc shift is introduced to break up any cycling. If 30 iterations are performed without convergence, the subroutine gives up. Although when this happens the matrix returned is not quasi-triangular, it is still almost exactly similar to the original matrix, and the similarity transformation has been accumulated in V.

EXCHNG works as follows. The first block is used to determine an implicit *QR* shift. An arbitrary *QR* step is performed on both blocks to eliminate the uncoupling between them. Then a sequence of *QR* steps using the previously determined shift

is performed on both blocks. Except in ill-conditioned cases, a block of size $B1$ having the eigenvalues of the first block will emerge in the lower part of the array occupied by both blocks, usually in one or two steps. If 30 iterations pass without convergence (the criterion is the same as in *HQR3*), the subroutine gives an error return.

Both *HQR3* and *EXCHNG* use the subroutine *QRSTEP* to perform the *QR* iterations. In addition, *HQR3* uses the subroutine *SPLIT* to separate real eigenvalues of a 2×2 block.

REFERENCES

1. Peters, G., and Wilkinson, J.H. Eigenvectors of real and complex matrices by LR and QR triangularizations. *Numer. Math. 16* (1970), 181–204.
2. Smith, B.T., Boyle, J.M., Garbow, B.S., Ikebe, Y., Klema, V.C., and Moler, C.B. *Matrix Eigensystem Routines—EISPACK Guide.* Lecture Notes in Computer Science, Vol. 6, Springer, New York, 1974.
3. Stewart, G.W. *Introduction to Matrix Computations.* Academic Press, New York, 1974.
4. Stewart, G.W. Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices. *Numer. Math. 25* (1976), 123–136.
5. Wilkinson, J.H. *The Algebraic Eigenvalue Problem.* Clarendon, New York, 1965.

ALGORITHM

```
      SUBROUTINE HQR3(A, V, N, NLOW, NUP, EPS, ER, EI, TYPE, NA, NV)    HQR   1Ø
      INTEGER N, NA, NLOW, NUP, NV, TYPE(N)                             HQR   2Ø
      REAL A(NA,N), EI(N), ER(N), EPS, V(NV,N)                          HQR   3Ø
C HQR3 REDUCES THE UPPER HESSENBERG MATRIX A TO QUASI-                  HQR   4Ø
C TRIANGULAR FORM BY UNITARY SIMILARITY TRANSFORMATIONS.                HQR   5Ø
C THE EIGENVALUES OF A, WHICH ARE CONTAINED IN THE 1X1                  HQR   6Ø
C AND 2X2 DIAGONAL BLOCKS OF THE REDUCED MATRIX, ARE                    HQR   7Ø
C ORDERED IN DESCENDING ORDER OF MAGNITUDE ALONG THE                    HQR   8Ø
C DIAGONAL.  THE TRANSFORMATIONS ARE ACCUMULATED IN THE                 HQR   9Ø
C ARRAY V.  HQR3 REQUIRES THE SUBROUTINES EXCHNG,                       HQR  1ØØ
C QRSTEP, AND SPLIT.  THE PARAMETERS IN THE CALLING                     HQR  11Ø
C SEQUENCE ARE (STARRED PARAMETERS ARE ALTERED BY THE                   HQR  12Ø
C SUBROUTINE)                                                           HQR  13Ø
C    *A       AN ARRAY THAT INITIALLY CONTAINS THE N X N               HQR  14Ø
C             UPPER HESSENBERG MATRIX TO BE REDUCED.  ON               HQR  15Ø
C             RETURN A CONTAINS THE REDUCED, QUASI-                    HQR  16Ø
C             TRIANGULAR MATRIX.                                       HQR  17Ø
C    *V       AN ARRAY THAT CONTAINS A MATRIX INTO WHICH               HQR  18Ø
C             THE REDUCING TRANSFORMATIONS ARE TO BE                   HQR  19Ø
C             MULTIPLIED.                                              HQR  2ØØ
C     N       THE ORDER OF THE MATRICES A AND V.                       HQR  21Ø
C     NLOW    A(NLOW,NLOW-1) AND A(NUP,+1,NUP) ARE                     HQR  22Ø
C     NUP     ASSUMED TO BE ZERO, AND ONLY ROWS NLOW                   HQR  23Ø
C             THROUGH NUP AND COLUMNS NLOW THROUGH                     HQR  24Ø
C             NUP ARE TRANSFORMED, RESULTING IN THE                    HQR  25Ø
C             CALCULATION OF EIGENVALUES NLOW                          HQR  26Ø
C             THROUGH NUP.                                             HQR  27Ø
C     EPS     A CONVERGENCE CRITERION.                                 HQR  28Ø
C    *ER      AN ARRAY THAT ON RETURN CONTAINS THE REAL                HQR  29Ø
C             PARTS OF THE EIGENVALUES.                                HQR  3ØØ
C    *EI      AN ARRAY THAT ON RETURN CONTAINS THE                     HQR  31Ø
C             IMAGINARY PARTS OF THE EIGENVALUES.                      HQR  32Ø
C    *TYPE    AN INTEGER ARRAY WHOSE I-TH ENTRY IS                     HQR  33Ø
C               Ø   IF THE I-TH EIGENVALUE IS REAL,                    HQR  34Ø
C               1   IF THE I-TH EIGENVALUE IS COMPLEX                  HQR  35Ø
C                   WITH POSITIVE IMAGINARY PART.                      HQR  36Ø
C               2   IF THE I-TH EIGENVALUE IS COMPLEX                  HQR  37Ø
C                   WITH NEGATIVE IMAGINARY PART,                      HQR  38Ø
C              -1   IF THE I-TH EIGENVALUE WAS NOT                     HQR  39Ø
C                   CALCULATED SUCCESSFULLY.                           HQR  4ØØ
C     NA      THE FIRST DIMENSION OF THE ARRAY A.                      HQR  41Ø
C     NV      THE FIRST DIMENSION OF THE ARRAY V.                      HQR  42Ø
C THE CONVERGENCE CRITERION EPS IS USED TO DETERMINE                    HQR  43Ø
C WHEN A SUBDIAGONAL ELEMENT OF A IS NEGLIGIBLE.                        HQR  44Ø
C SPECIFICALLY A(I+1,I) IS REGARDED AS NEGLIGIBLE                       HQR  45Ø
C IF                                                                    HQR  46Ø
C       ABS(A(I+1),I)) .LE. EPS*(ABS(A(I,I))+ABS(A(I+1,I+1))).         HQR  47Ø
C THIS MEANS THAT THE FINAL MATRIX RETURNED BY THE                      HQR  48Ø
```

```
C PROGRAM WILL BE EXACTLY SIMILAR TO A + E WHERE E IS                    HQR  49Ø
C OF ORDER EPS*NORM(A), FOR ANY REASONABLY BALANCED NORM                 HQR  5ØØ
C SUCH AS THE ROW-SUM NORM.                                              HQR  51Ø
C INTERNAL VARIABLES                                                     HQR  52Ø
      INTEGER I, IT, L, MU, NL, NU                                       HQR  53Ø
      REAL E1, E2, P, Q, R, S, T, W, X, Y, Z                             HQR  54Ø
      LOGICAL FAIL                                                       HQR  55Ø
C INITIALIZE.                                                            HQR  56Ø
      DO 1Ø I=NLOW,NUP                                                   HQR  57Ø
        TYPE(I) = -1                                                     HQR  58Ø
   1Ø CONTINUE                                                           HQR  59Ø
      T = Ø.                                                             HQR  6ØØ
C MAIN LOOP. FIND AND ORDER EIGENVALUES.                                 HQR  61Ø
      NU = NUP                                                           HQR  62Ø
   2Ø IF (NU.LT.NLOW) GO TO 24Ø                                          HQR  63Ø
      IT = Ø                                                             HQR  64Ø
C QR LOOP.  FIND NEGLIGIBLE ELEMENTS AND PERFORM                         HQR  65Ø
C QR STEPS.                                                              HQR  66Ø
   3Ø CONTINUE                                                           HQR  67Ø
C SEARCH BACK FOR NEGLIGIBLE ELEMENTS.                                   HQR  68Ø
      L = NU                                                             HQR  69Ø
   4Ø CONTINUE                                                           HQR  7ØØ
      IF (L.EQ.NLOW) GO TO 5Ø                                            HQR  71Ø
      IF (ABS(A(L,L-1)).LT.EPS*(ABS(A(L-1,L-1))+ABS(A(L,L)))) GO TO      HQR  72Ø
     * 5Ø                                                                HQR  73Ø
      L = L - 1                                                          HQR  74Ø
      GO TO 4Ø                                                           HQR  75Ø
   5Ø CONTINUE                                                           HQR  76Ø
C TEST TO SEE IF AN EIGENVALUE OR A 2X2 BLOCK                            HQR  77Ø
C HAS BEEN FOUND.                                                        HQR  78Ø
      X = A(NU,NU)                                                       HQR  79Ø
      IF (L.EQ.NU) GO TO 16Ø                                             HQR  8ØØ
      Y = A(NU-1,NU-1)                                                   HQR  81Ø
      W = A(NU,NU-1)*A(NU-1,NU)                                          HQR  82Ø
      IF (L.EQ.NU-1) GO TO 1ØØ                                           HQR  83Ø
C TEST ITERATION COUNT. IF IT IS 3Ø QUIT.  IF                            HQR  84Ø
C IT IS 1Ø OR 2Ø SET UP AN AD-HOC SHIFT.                                 HQR  85Ø
      IF (IT.EQ.3Ø) GO TO 24Ø                                            HQR  86Ø
      IF (IT.NE.1Ø .AND. IT.NE.2Ø) GO TO 7Ø                              HQR  87Ø
C AD-HOC SHIFT.                                                          HQR  88Ø
      T = T + X                                                          HQR  89Ø
      DO 6Ø I=NLOW,NU                                                    HQR  9ØØ
        A(I,I) = A(I,I) - X                                              HQR  91Ø
   6Ø CONTINUE                                                           HQR  92Ø
      S = ABS(A(NU,NU-1)) + ABS(A(NU-1,NU-2))                            HQR  93Ø
      X = Ø.75*S                                                         HQR  94Ø
      Y = X                                                              HQR  95Ø
      W = -Ø.4375*S**2                                                   HQR  96Ø
   7Ø CONTINUE                                                           HQR  97Ø
      IT = IT + 1                                                        HQR  98Ø
C LOOK FOR TWO CONSECUTIVE SMALL SUB-DIAGONAL                            HQR  99Ø
C ELEMENTS.                                                              HQR 1ØØØ
      NL = NU - 2                                                        HQR 1Ø1Ø
   8Ø CONTINUE                                                           HQR 1Ø2Ø
      Z = A(NL,NL)                                                       HQR 1Ø3Ø
      R = X - Z                                                          HQR 1Ø4Ø
      S = Y - Z                                                          HQR 1Ø5Ø
      P = (R*S-W)/A(NL+1,NL) + A(NL,NL+1)                                HQR 1Ø6Ø
      Q = A(NL+1,NL+1) - Z - R - S                                       HQR 1Ø7Ø
      R = A(NL+2,NL+1)                                                   HQR 1Ø8Ø
      S = ABS(P) + ABS(Q) + ABS(R)                                       HQR 1Ø9Ø
      P = P/S                                                            HQR 11ØØ
      Q = Q/S                                                            HQR 111Ø
      R = R/S                                                            HQR 112Ø
      IF (NL.EQ.L) GO TO 9Ø                                              HQR 113Ø
      IF (ABS(A(NL,NL-1))*(ABS(Q)+ABS(R)).LE.EPS*ABS(P)*(ABS(A(NL-1,     HQR 114Ø
     * NL-1))+ABS(Z)+ABS(A(NL+1,NL+1)))) GO TO 9Ø                        HQR 115Ø
      NL = NL - 1                                                        HQR 116Ø
      GO TO 8Ø                                                           HQR 117Ø
   9Ø CONTINUE                                                           HQR 118Ø
C PERFORM A QR STEP BETWEEN NL AND NU.                                   HQR 119Ø
      CALL QRSTEP(A, V, P, Q, R, NL, NU, N, NA, NV)                      HQR 12ØØ
      GO TO 3Ø                                                           HQR 121Ø
C 2X2 BLOCK FOUND.                                                       HQR 122Ø
  1ØØ IF (NU.NE.NLOW+1) A(NU-1,NU-2) = Ø.                                HQR 123Ø
      A(NU,NU) = A(NU,NU) + T                                            HQR 124Ø
```

```
      A(NU-1,NU-1) = A(NU-1,NU-1) + T                                   HQR 125Ø
      TYPE(NU) = Ø                                                      HQR 126Ø
      TYPE(NU-1) = Ø                                                    HQR 127Ø
      MU = NU                                                           HQR 128Ø
C LOOP TO POSITION  2X2 BLOCK.                                          HQR 129Ø
  11Ø CONTINUE                                                          HQR 13ØØ
      NL = MU - 1                                                       HQR 131Ø
C ATTEMPT  TO SPLIT THE BLOCK INTO TWO REAL                             HQR 132Ø
C EIGENVALUES.                                                          HQR 133Ø
      CALL SPLIT(A, V, N, NL, E1, E2, NA, NV)                           HQR 134Ø
C IF THE SPLIT WAS SUCCESSFUL, GO AND ORDER THE                         HQR 135Ø
C REAL EIGENVALUES.                                                     HQR 136Ø
      IF (A(MU,MU-1).EQ.Ø.) GO TO 17Ø                                   HQR 137Ø
C TEST TO SEE IF THE BLOCK IS PROPERLY POSITIONED,                      HQR 138Ø
C AND IF NOT EXCHANGE IT                                                HQR 139Ø
      IF (MU.EQ.NUP) GO TO 23Ø                                          HQR 14ØØ
      IF (MU.EQ.NUP-1) GO TO 13Ø                                        HQR 141Ø
      IF (A(MU+2,MU+1).EQ.Ø.) GO TO 13Ø                                 HQR 142Ø
C THE NEXT BLOCK IS 2X2.                                                HQR 143Ø
      IF (A(MU-1,MU-1)*A(MU,MU)-A(MU-1,MU)*A(MU,MU-1).GE.A(MU+1,        HQR 144Ø
     * MU+1)*A(MU+2,MU+2)-A(MU+1,MU+2)*A(MU+2,MU+1)) GO TO 23Ø           HQR 145Ø
      CALL EXCHNG(A, V, N, NL, 2, 2, EPS, FAIL, NA, NV)                 HQR 146Ø
      IF (.NOT.FAIL) GO TO 12Ø                                          HQR 147Ø
      TYPE(NL) = -1                                                     HQR 148Ø
      TYPE(NL+1) = -1                                                   HQR 149Ø
      TYPE(NL+2) = -1                                                   HQR 15ØØ
      TYPE(NL+3) = -1                                                   HQR 151Ø
      GO TO 24Ø                                                         HQR 152Ø
  12Ø CONTINUE                                                          HQR 153Ø
      MU = MU + 2                                                       HQR 154Ø
      GO TO 15Ø                                                         HQR 155Ø
  13Ø CONTINUE                                                          HQR 156Ø
C THE NEXT BLOCK IS 1X1.                                                HQR 157Ø
      IF (A(MU-1,MU-1)*A(MU,MU)-A(MU-1,MU)*A(MU,MU-1).GE.A(MU+1,        HQR 158Ø
     * MU+1)**2) GO TO 23Ø                                              HQR 159Ø
      CALL EXCHNG(A, V, N, NL, 2, 1, EPS, FAIL, NA, NV)                 HQR 16ØØ
      IF (.NOT.FAIL) GO TO 14Ø                                          HQR 161Ø
      TYPE(NL) = -1                                                     HQR 162Ø
      TYPE(NL+1) = -1                                                   HQR 163Ø
      TYPE(NL+2) = -1                                                   HQR 164Ø
      GO TO 24Ø                                                         HQR 165Ø
  14Ø CONTINUE                                                          HQR 166Ø
      MU = MU + 1                                                       HQR 167Ø
  15Ø CONTINUE                                                          HQR 168Ø
      GO TO 11Ø                                                         HQR 169Ø
C SINGLE EIGENVALUE FOUND.                                              HQR 17ØØ
  16Ø NL = Ø                                                            HQR 171Ø
      A(NU,NU) = A(NU,NU) + T                                           HQR 172Ø
      IF (NU.NE.NLOW) A(NU,NU-1) = Ø.                                   HQR 173Ø
      TYPE(NU) = Ø                                                      HQR 174Ø
      MU = NU                                                           HQR 175Ø
C LOOP TO POSITION ONE OR TWO REAL EIGENVALUES.                         HQR 176Ø
  17Ø CONTINUE                                                          HQR 177Ø
C POSITION THE EIGENVALUE LOCATED AT A(NL,NL).                          HQR 178Ø
  18Ø CONTINUE                                                          HQR 179Ø
      IF (MU.EQ.NUP) GO TO 22Ø                                          HQR 18ØØ
      IF (MU.EQ.NUP-1) GO TO 2ØØ                                        HQR 181Ø
      IF (A(MU+2,MU+1).EQ.Ø.) GO TO 2ØØ                                 HQR 182Ø
C THE NEXT BLOCK IS 2X2.                                                HQR 183Ø
      IF (A(MU,MU)**2.GE.A(MU+1,MU+1)*A(MU+2,MU+2)-A(MU+1,MU+2)*        HQR 184Ø
     * A(MU+2,MU+1)) GO TO 23Ø                                          HQR 185Ø
      CALL EXCHNG(A, V, N, MU, 1, 2, EPS, FAIL, NA, NV)                 HQR 186Ø
      IF (.NOT.FAIL) GO TO 19Ø                                          HQR 187Ø
      TYPE(MU) = -1                                                     HQR 188Ø
      TYPE(MU+1) = -1                                                   HQR 189Ø
      TYPE(MU+2) = -1                                                   HQR 19ØØ
      GO TO 24Ø                                                         HQR 191Ø
  19Ø CONTINUE                                                          HQR 192Ø
      MU = MU + 2                                                       HQR 193Ø
      GO TO 21Ø                                                         HQR 194Ø
  2ØØ CONTINUE                                                          HQR 195Ø
C THE NEXT BLOCK IS 1X1.                                                HQR 196Ø
      IF (ABS(A(MU,MU)).GE.ABS(A(MU+1,MU+1))) GO TO 22Ø                 HQR 197Ø
      CALL EXCHNG(A, V, N, MU, 1, 1, EPS, FAIL, NA, NV)                 HQR 198Ø
      MU = MU + 1                                                       HQR 199Ø
  21Ø CONTINUE                                                          HQR 2ØØØ
```

```
      GO TO 18Ø                                                    HQR 2Ø1Ø
   22Ø CONTINUE                                                    HQR 2Ø2Ø
      MU = NL                                                      HQR 2Ø3Ø
      NL = Ø                                                       HQR 2Ø4Ø
      IF (MU.NE.Ø) GO TO 17Ø                                       HQR 2Ø5Ø
C GO BACK AND GET THE NEXT EIGENVALUE.                             HQR 2Ø6Ø
   23Ø CONTINUE                                                    HQR 2Ø7Ø
      NU = L - 1                                                   HQR 2Ø8Ø
      GO TO 2Ø                                                     HQR 2Ø9Ø
C ALL THE EIGENVALUES HAVE BEEN FOUND AND ORDERED.                 HQR 21ØØ
C COMPUTE THEIR VALUES AND TYPE.                                   HQR 211Ø
   24Ø IF (NU.LT.NLOW) GO TO 26Ø                                   HQR 212Ø
      DO 25Ø I=1,NU                                                HQR 213Ø
        A(I,I) = A(I,I) + T                                        HQR 214Ø
   25Ø CONTINUE                                                    HQR 215Ø
   26Ø CONTINUE                                                    HQR 216Ø
      NU = NUP                                                     HQR 217Ø
   27Ø CONTINUE                                                    HQR 218Ø
      IF (TYPE(NU).NE.-1) GO TO 28Ø                                HQR 219Ø
      NU = NU - 1                                                  HQR 22ØØ
      GO TO 31Ø                                                    HQR 221Ø
   28Ø CONTINUE                                                    HQR 222Ø
      IF (NU.EQ.NLOW) GO TO 29Ø                                    HQR 223Ø
      IF (A(NU,NU-1).EQ.Ø.) GO TO 29Ø                              HQR 224Ø
C 2X2 BLOCK.                                                       HQR 225Ø
      CALL SPLIT(A, V, N, NU-1, E1, E2, NA, NV)                    HQR 226Ø
      IF (A(NU,NU-1).EQ.Ø.) GO TO 29Ø                              HQR 227Ø
      ER(NU) = E1                                                  HQR 228Ø
      EI(NU-1) = E2                                                HQR 229Ø
      ER(NU-1) = ER(NU)                                            HQR 23ØØ
      EI(NU) = -EI(NU-1)                                           HQR 231Ø
      TYPE(NU-1) = 1                                               HQR 232Ø
      TYPE(NU) = 2                                                 HQR 233Ø
      NU = NU - 2                                                  HQR 234Ø
      GO TO 3ØØ                                                    HQR 235Ø
   29Ø CONTINUE                                                    HQR 236Ø
C SINGLE ROOT.                                                     HQR 237Ø
      ER(NU) = A(NU,NU)                                            HQR 238Ø
      EI(NU) = Ø.                                                  HQR 239Ø
      NU = NU - 1                                                  HQR 24ØØ
   3ØØ CONTINUE                                                    HQR 241Ø
   31Ø CONTINUE                                                    HQR 242Ø
      IF (NU.GE.NLOW) GO TO 27Ø                                    HQR 243Ø
      RETURN                                                       HQR 244Ø
      END                                                          HQR 245Ø


      SUBROUTINE SPLIT(A, V, N, L, E1, E2, NA, NV)                 SPL  1Ø
      INTEGER L, N, NA, NV                                         SPL  2Ø
      REAL A(NA,N), V(NV,N)                                        SPL  3Ø
C GIVEN THE UPPER HESSENBERG MATRIX A WITH A 2X2 BLOCK             SPL  4Ø
C STARTING AT A(L,L), SPLIT DETERMINES IF THE                      SPL  5Ø
C CORRESPONDING EIGENVALUES ARE REAL OR COMPLEX. IF THEY           SPL  6Ø
C ARE REAL, A ROTATION IS DETERMINED THAT REDUCES THE              SPL  7Ø
C BLOCK TO UPPER TRIANGULAR FORM WITH THE EIGENVALUE               SPL  8Ø
C OF LARGEST ABSOLUTE VALUE APPEARING FIRST.  THE                  SPL  9Ø
C ROTATION IS ACCUMULATED IN V.  THE EIGENVALUES (REAL             SPL  1ØØ
C OR COMPLEX) ARE RETURNED IN E1 AND E2.  THE PARAMETERS           SPL  11Ø
C IN THE CALLING SEQUENCE ARE (STARRED PARAMETERS ARE              SPL  12Ø
C ALTERED BY THE SUBROUTINE)                                       SPL  13Ø
C    *A       THE UPPER HESSENVERG MATRIX WHOSE 2X2               SPL  14Ø
C             BLOCK IS TO BE SPLIT.                               SPL  15Ø
C    *V       THE ARRAY IN WHICH THE SPLITTING TRANS-             SPL  16Ø
C             FORMATION IS TO BE ACCUMULATED.                     SPL  17Ø
C     N       THE ORDER OF THE MATRIX A.                          SPL  18Ø
C     L       THE POSITION OF THE 2X2 BLOCK.                      SPL  19Ø
C    *E1      ON RETURN IF THE EIGENVALUES ARE COMPLEX            SPL  2ØØ
C    *E2      E1 CONTAINS THEIR COMMON REAL PART AND              SPL  21Ø
C             E2 CONTAINS THE POSITIVE IMAGINARY PART.            SPL  22Ø
C             IF THE EIGENVALUES ARE REAL, E1 CONTAINS            SPL  23Ø
C             THE ONE LARGEST IN ABSOLUTE VALUE AND E2            SPL  24Ø
C             CONTAINS THE OTHER ONE.                             SPL  25Ø
C     NA      THE FIRST DIMENSION OF THE ARRAY A.                 SPL  26Ø
C     NV      THE FIRST DIMENSION OF THE ARRAY V.                 SPL  27Ø
C INTERNAL VARIABLES                                               SPL  28Ø
      INTEGER I, J, L1                                             SPL  29Ø
```

```
      REAL P, Q, R, T, U, W, X, Y, Z                                    SPL  3ØØ
      X = A(L+1,L+1)                                                    SPL  31Ø
      Y = A(L,L)                                                        SPL  32Ø
      W = A(L,L+1)*A(L+1,L)                                             SPL  33Ø
      P = (Y-X)/2.                                                      SPL  34Ø
      Q = P**2 + W                                                      SPL  35Ø
      IF (Q.GE.Ø.) GO TO 1Ø                                             SPL  36Ø
C COMPLEX EIGENVALUE.                                                   SPL  37Ø
      E1 = P + X                                                        SPL  38Ø
      E2 = SQRT(-Q)                                                     SPL  39Ø
      RETURN                                                            SPL  4ØØ
   1Ø CONTINUE                                                          SPL  41Ø
C TWO REAL EIGENVALUES.  SET UP TRANSFORMATION.                         SPL  42Ø
      Z = SQRT(Q)                                                       SPL  43Ø
      IF (P.LT.Ø.) GO TO 2Ø                                             SPL  44Ø
      Z = P + Z                                                         SPL  45Ø
      GO TO 3Ø                                                          SPL  46Ø
   2Ø CONTINUE                                                          SPL  47Ø
      Z = P - Z                                                         SPL  48Ø
   3Ø CONTINUE                                                          SPL  49Ø
      IF (Z.EQ.Ø.) GO TO 4Ø                                             SPL  5ØØ
      R = -W/Z                                                          SPL  51Ø
      GO TO 5Ø                                                          SPL  52Ø
   4Ø CONTINUE                                                          SPL  53Ø
      R = Ø.                                                            SPL  54Ø
   5Ø CONTINUE                                                          SPL  55Ø
      IF (ABS(X+Z).GE.ABS(X+R)) Z = R                                   SPL  56Ø
      Y = Y - X - Z                                                     SPL  57Ø
      X = -Z                                                            SPL  58Ø
      T = A(L,L+1)                                                      SPL  59Ø
      U = A(L+1,L)                                                      SPL  6ØØ
      IF (ABS(Y)+ABS(U).LE.ABS(T)+ABS(X)) GO TO 6Ø                      SPL  61Ø
      Q = U                                                             SPL  62Ø
      P = Y                                                             SPL  63Ø
      GO TO 7Ø                                                          SPL  64Ø
   6Ø CONTINUE                                                          SPL  65Ø
      Q = X                                                             SPL  66Ø
      P = T                                                             SPL  67Ø
   7Ø CONTINUE                                                          SPL  68Ø
      R = SQRT(P**2+Q**2)                                               SPL  69Ø
      IF (R.GT.Ø.) GO TO 8Ø                                             SPL  7ØØ
      E1 = A(L,L)                                                       SPL  71Ø
      E2 = A(L+1,L+1)                                                   SPL  72Ø
      A(L+1,L) = Ø.                                                     SPL  73Ø
      RETURN                                                            SPL  74Ø
   8Ø CONTINUE                                                          SPL  75Ø
      P = P/R                                                           SPL  76Ø
      Q = Q/R                                                           SPL  77Ø
C PREMULTIPLY.                                                          SPL  78Ø
      DO 9Ø J=L,N                                                       SPL  79Ø
        Z = A(L,J)                                                      SPL  8ØØ
        A(L,J) = P*Z + Q*A(L+1,J)                                       SPL  81Ø
        A(L+1,J) = P*A(L+1,J) - Q*Z                                     SPL  82Ø
   9Ø CONTINUE                                                          SPL  83Ø
C POSTMULTIPLY.                                                         SPL  84Ø
      L1 = L + 1                                                        SPL  85Ø
      DO 1ØØ I=1,L1                                                     SPL  86Ø
        Z = A(I,L)                                                      SPL  87Ø
        A(I,L) = P*Z + Q*A(I,L+1)                                       SPL  88Ø
        A(I,L+1) = P*A(I,L+1) - Q*Z                                     SPL  89Ø
  1ØØ CONTINUE                                                          SPL  9ØØ
C ACCUMULATE THE TRANSFORMATION IN V.                                   SPL  91Ø
      DO 11Ø I=1,N                                                      SPL  92Ø
        Z = V(I,L)                                                      SPL  93Ø
        V(I,L) = P*Z + Q*V(I,L+1)                                       SPL  94Ø
        V(I,L+1) = P*V(I,L+1) - Q*Z                                     SPL  95Ø
  11Ø CONTINUE                                                          SPL  96Ø
      A(L+1,L) = Ø.                                                     SPL  97Ø
      E1 = A(L,L)                                                       SPL  98Ø
      E2 = A(L+1,L+1)                                                   SPL  99Ø
      RETURN                                                            SPL 1ØØØ
      END                                                               SPL 1Ø1Ø


       SUBROUTINE (EXCHNG(A, V, N, L, B1, B2, EPS, FAIL, NA, NV)        EXC   1Ø
       INTEGER B1, B2, L, NA, NV                                        EXC   2Ø
```

```
      REAL A(NA,N), EPS, V(NV,N)                                        EXC  3Ø
      LOGICAL FAIL                                                      EXC  4Ø
C GIVEN THE UPPER HESSENBERG MATRIX A WITH CONSECUTIVE                  EXC  5Ø
C B1XB1 AND B2XB2 DIAGONAL BLOCKS (B1,B2 .LE. 2)                        EXC  6Ø
C STARTING AT A(L,L), EXCHNG PRODUCES A UNITARY                         EXC  7Ø
C SIMILARITY TRANSFORMATION THAT EXCHANGES THE BLOCKS                   EXC  8Ø
C ALONG WITH THEIR EIGENVALUES.  THE TRANSFORMATION                     EXC  9Ø
C IS ACCUMULATED IN V.  EXCHNG REQUIRES THE SUBROUTINE                  EXC  1ØØ
C QRSTEP.  THE PARAMETERS IN THE CALLING SEQUENCE ARE                   EXC  11Ø
C (STARRED PARAMETERS ARE ALTERED BY THE SUBROUTINE)                    EXC  12Ø
C    *A       THE MATRIX WHOSE BLOCKS ARE TO BE                         EXC  13Ø
C             INTERCHANGED.                                             EXC  14Ø
C    *V       THE ARRAY INTO WHICH THE TRANSFORMATIONS                  EXC  15Ø
C             ARE TO BE ACCUMULATED.                                    EXC  16Ø
C     N       THE ORDER OF THE MATRIX A.                                EXC  17Ø
C     L       THE POSITION OF THE BLOCKS.                               EXC  18Ø
C     B1      AN INTEGER CONTAINING THE SIZE OF THE                     EXC  19Ø
C             FIRST BLOCK.                                              EXC  2ØØ
C     B2      AN INTEGER CONTAINING THE SIZE OF THE                     EXC  21Ø
C             SECOND BLOCK.                                             EXC  22Ø
C     EPS     A CONVERGENCE CRITERION (CF. HQR3).                       EXC  23Ø
C    *FAIL    A LOGICAL VARIABLE WHICH IS FALSE ON A                    EXC  24Ø
C             NORMAL RETURN.  IF THIRTY ITERATIONS WERE                 EXC  25Ø
C             PERFORMED WITHOUT CONVERGENCE, FAIL IS SET                EXC  26Ø
C             TO TRUE AND THE ELEMENT                                   EXC  27Ø
C             A(L+B2,L+B2-1) CANNOT BE ASSUMED ZERO.                    EXC  28Ø
C     NA      THE FIRST DIMENSION OF THE ARRAY A.                       EXC  29Ø
C     NV      THE FIRST DIMENSION OF THE ARRAY V.                       EXC  3ØØ
C INTERNAL VARIABLES.                                                   EXC  31Ø
      INTEGER I, IT, J, L1, M                                           EXC  32Ø
      REAL P, Q, R, S, W, X, Y, Z                                       EXC  33Ø
      FAIL = .FALSE.                                                    EXC  34Ø
      IF (B1.EQ.2) GO TO 7Ø                                             EXC  35Ø
      IF (B2.EQ.2) GO TO 4Ø                                             EXC  36Ø
C INTERCHANGE 1X1 AND 1X1 BLOCKS.                                       EXC  37Ø
      L1 = L + 1                                                        EXC  38Ø
      Q = A(L+1,L+1) - A(L,L)                                           EXC  39Ø
      P = A(L,L+1)                                                      EXC  4ØØ
      R = AMAX1(P,Q)                                                    EXC  41Ø
      IF (R.EQ.Ø.) RETURN                                               EXC  42Ø
      P = P/R                                                           EXC  43Ø
      Q = Q/R                                                           EXC  44Ø
      R = SQRT(P**2+Q**2)                                               EXC  45Ø
      P = P/R                                                           EXC  46Ø
      Q = Q/R                                                           EXC  47Ø
      DO 1Ø J=L,N                                                       EXC  48Ø
        S = P*A(L,J) + Q*A(L+1,J)                                       EXC  49Ø
        A(L+1,J) = P*A(L+1,J) - Q*A(L,J)                                EXC  5ØØ
        A(L,J) = S                                                      EXC  51Ø
   1Ø CONTINUE                                                          EXC  52Ø
      DO 2Ø I=1,L1                                                      EXC  53Ø
        S = P*A(I,L) + Q*A(I,L+1)                                       EXC  54Ø
        A(I,L+1) = P*A(I,L+1) - Q*A(I,L)                                EXC  55Ø
        A(I,L) = S                                                      EXC  56Ø
   2Ø CONTINUE                                                          EXC  57Ø
      DO 3Ø I=1,N                                                       EXC  58Ø
        S = P*V(I,L) + Q*V(I,L+1)                                       EXC  59Ø
        V(I,L+1) = P*V(I,L+1) - Q*V(I,L)                                EXC  6ØØ
        V(I,L) = S                                                      EXC  61Ø
   3Ø CONTINUE                                                          EXC  62Ø
      A(L+1,L) = Ø.                                                     EXC  63Ø
      RETURN                                                            EXC  64Ø
   4Ø CONTINUE                                                          EXC  65Ø
C INTERCHANGE 1X1 AND 2X2 BLOCKS.                                       EXC  66Ø
      X = A(L,L)                                                        EXC  67Ø
      P = 1.                                                            EXC  68Ø
      Q = 1.                                                            EXC  69Ø
      R = 1.                                                            EXC  7ØØ
      CALL QRSTEP(A, V, P, Q, R, L, L+2, N, NA, NV)                     EXC  71Ø
      IT = Ø                                                            EXC  72Ø
   5Ø IT = IT + 1                                                       EXC  73Ø
      IF (IT.LE.3Ø) GO TO 6Ø                                            EXC  74Ø
      FAIL = .TRUE.                                                     EXC  75Ø
      RETURN                                                            EXC  76Ø
   6Ø CONTINUE                                                          EXC  77Ø
```

```
      P = A(L,L) - X                                                    EXC  78Ø
      Q = A(L+1,L)                                                      EXC  79Ø
      R = Ø.                                                            EXC  8ØØ
      CALL QRSTEP(A, V, P, Q, R, L, L+2, N, NA, NV)                     EXC  81Ø
      IF (ABS(A(L+2,L+1)).GT.EPS*(ABS(A(L+1,L+1))+ABS(A(L+2,L+2))))     EXC  82Ø
     * GO TO 5Ø                                                         EXC  83Ø
      A(L+2,L+1) = Ø.                                                   EXC  84Ø
      RETURN                                                            EXC  85Ø
   7Ø CONTINUE                                                          EXC  86Ø
C INTERCHANGE 2X2 AND B2XB2 BLOCKS.                                     EXC  87Ø
      M = L + 2                                                         EXC  88Ø
      IF (B2.EQ.2) M = M + 1                                            EXC  89Ø
      X = A(L+1,L+1)                                                    EXC  9ØØ
      Y = A(L,L)                                                        EXC  91Ø
      W = A(L+1,L)*A(L,L+1)                                             EXC  92Ø
      P = 1.                                                            EXC  93Ø
      Q = 1.                                                            EXC  94Ø
      R = 1.                                                            EXC  95Ø
      CALL QRSTEP(A, V, P, Q, R, L, M, N, NA, NV)                       EXC  96Ø
      IT = Ø                                                            EXC  97Ø
   8Ø IT = IT + 1                                                       EXC  98Ø
      IF (IT.LE.3Ø) GO TO 9Ø                                            EXC  99Ø
      FAIL = .TRUE.                                                     EXC 1ØØØ
      RETURN                                                            EXC 1Ø1Ø
   9Ø CONTINUE                                                          EXC 1Ø2Ø
      Z = A(L,L)                                                        EXC 1Ø3Ø
      R = X - Z                                                         EXC 1Ø4Ø
      S = Y - Z                                                         EXC 1Ø5Ø
      P = (R*S-W)/A(L+1,L) + A(L,L+1)                                   EXC 1Ø6Ø
      Q = A(L+1,L+1) - Z - R - S                                        EXC 1Ø7Ø
      R = A(L+2,L+1)                                                    EXC 1Ø8Ø
      S = ABS(P) + ABS(Q) + ABS(R)                                      EXC 1Ø9Ø
      P = P/S                                                           EXC 11ØØ
      Q = Q/S                                                           EXC 111Ø
      R = R/S                                                           EXC 112Ø
      CALL QRSTEP(A, V, P, Q, R, L, M, N, NA, NV)                       EXC 113Ø
      IF (ABS(A(M-1,M-2)).GT.EPS*(ABS(A(M-1,M-1))+ABS(A(M-2,M-2))))     EXC 114Ø
     * GO TO 8Ø                                                         EXC 115Ø
      A(M-1,M-2) = Ø.                                                   EXC 116Ø
      RETURN                                                            EXC 117Ø
      END                                                               EXC 118Ø



      SUBROUTINE QRSTEP(A, V, P, Q, R, NL, NU, N, NA, NV)               QRS   1Ø
      INTEGER N, NA, NL, NU, NV                                         QRS   2Ø
      REAL A(NA,N), P, Q, R, V(NV,N)                                    QRS   3Ø
C QRSTEP PERFORMS ONE IMPLICIT QR STEP ON THE                           QRS   4Ø
C UPPER HESSENBERG MATRIX A.  THE SHIFT IS DETERMINED                   QRS   5Ø
C BY THE NUMBERS P,Q, AND R, AND THE STEP IS APPLIED TO                 QRS   6Ø
C ROWS AND COLUMNS NL THROUGH NU.  THE TRANSFORMATIONS                  QRS   7Ø
C ARE ACCUMULATED IN V.  THE PARAMETERS IN THE CALLING                  QRS   8Ø
C SEQUENCE ARE (STARRED APRAMETERS ARE ALTERED BY THE                   QRS   9Ø
C SUBROUTINE)                                                           QRS  1ØØ
C    *A       THE UPPER HESSENBERG MATRIX ON WHICH THE                  QRS  11Ø
C             QR STEP IS TO BE PERFORMED.                               QRS  12Ø
C    *V       THE ARRAY IN WHICH THE TRANSFORMATIONS                    QRS  13Ø
C             ARE TO BE ACCUMULATED                                     QRS  14Ø
C    *P       PARAMETERS THAT DETERMINE THE SHIFT.                      QRS  15Ø
C    *Q                                                                 QRS  16Ø
C    *R                                                                 QRS  17Ø
C     NL      THE LOWER LIMIT OF THE STEP.                              QRS  18Ø
C     NU      THE UPPER LIMIT OF THE STEP.                              QRS  19Ø
C     N       THE ORDER OF THE MATRIX A.                                QRS  2ØØ
C     NA      THE FIRST DIMENSION OF THE ARRAY A.                       QRS  21Ø
C     NV      THE FIRST DIMENSION OF THE ARRAY V.                       QRS  22Ø
C INTERNAL VARIABLES.                                                   QRS  23Ø
      INTEGER I, J, K, NL2, NL3, NUM1                                   QRS  24Ø
      REAL S, X, Y, Z                                                   QRS  25Ø
      LOGICAL LAST                                                      QRS  26Ø
      NL2 = NL + 2                                                      QRS  27Ø
      DO 1Ø I=NL2,NU                                                    QRS  28Ø
        A(I,I-2) = Ø.                                                   QRS  29Ø
   1Ø CONTINUE                                                          QRS  3ØØ
      IF (NL2.EQ.NU) GO TO 3Ø                                           QRS  31Ø
      NL3 = NL + 3                                                      QRS  32Ø
```

```
      DO 2Ø I=NL3,NU                                                    QRS  33Ø
        A(I,I-3) = Ø.                                                   QRS  34Ø
   2Ø CONTINUE                                                          QRS  35Ø
   3Ø CONTINUE                                                          QRS  36Ø
      NUM1 = NU - 1                                                     QRS  37Ø
      DO 13Ø K=NL,NUM1                                                  QRS  38Ø
C DETERMINE THE TRANSFORMATION.                                         QRS  39Ø
        LAST = K.EQ.NUM1                                                QRS  4ØØ
        IF (K.EQ.NL) GO TO 4Ø                                           QRS  41Ø
        P = A(K,K-1)                                                    QRS  42Ø
        Q = A(K+1,K-1)                                                  QRS  43Ø
        R = Ø.                                                          QRS  44Ø
        IF (.NOT.LAST) R = A(K+2,K-1)                                   QRS  45Ø
        X = ABS(P) + ABS(Q) + ABS(R)                                    QRS  46Ø
        IF (X.EQ.Ø.) GO TO 13Ø                                          QRS  47Ø
        P = P/X                                                         QRS  48Ø
        Q = Q/X                                                         QRS  49Ø
        R = R/X                                                         QRS  5ØØ
   4Ø   CONTINUE                                                        QRS  51Ø
        S = SQRT(P**2+Q**2+R**2)                                        QRS  52Ø
        IF (P.LT.Ø.) S = -S                                             QRS  53Ø
        IF (K.EQ.NL) GO TO 5Ø                                           QRS  54Ø
        A(K,K-1) = -S*X                                                 QRS  55Ø
        GO TO 6Ø                                                        QRS  56Ø
   5Ø   CONTINUE                                                        QRS  57Ø
        IF (NL.NE.1) A(K,K-1) = -A(K,K-1)                               QRS  58Ø
   6Ø   CONTINUE                                                        QRS  59Ø
        P = P + S                                                       QRS  6ØØ
        X = P/S                                                         QRS  61Ø
        Y = Q/S                                                         QRS  62Ø
        Z = R/S                                                         QRS  63Ø
        Q = Q/P                                                         QRS  64Ø
        R = R/P                                                         QRS  65Ø
C PREMULTIPLY.                                                          QRS  66Ø
        DO 8Ø J=K,N                                                     QRS  67Ø
          P = A(K,J) + Q*A(K+1,J)                                       QRS  68Ø
          IF (LAST) GO TO 7Ø                                            QRS  69Ø
          P = P + R*A(K+2,J)                                            QRS  7ØØ
          A(K+2,J) = A(K+2,J) - P*Z                                     QRS  71Ø
   7Ø     CONTINUE                                                      QRS  72Ø
          A(K+1,J) = A(K+1,J) - P*Y                                     QRS  73Ø
          A(K,J) = A(K,J) - P*X                                         QRS  74Ø
   8Ø   CONTINUE                                                        QRS  75Ø
C POSTMULTIPLY.                                                         QRS  76Ø
        J = MINØ(K+3,NU)                                                QRS  77Ø
        DO 1ØØ I=1,J                                                    QRS  78Ø
          P = X*A(I,K) + Y*A(I,K+1)                                     QRS  79Ø
          IF (LAST) GO TO 9Ø                                            QRS  8ØØ
          P = P + Z*A(I,K+2)                                            QRS  81Ø
          A(I,K+2) = A(I,K+2) - P*R                                     QRS  82Ø
   9Ø     CONTINUE                                                      QRS  83Ø
          A(I,K+1) = A(I,K+1) - P*Q                                     QRS  84Ø
          A(I,K) = A(I,K) - P                                           QRS  85Ø
  1ØØ   CONTINUE                                                        QRS  86Ø
C ACCUMULATE THE TRANSFORMATION IN V.                                   QRS  87Ø
        DO 12Ø I=1,N                                                    QRS  88Ø
          P = X*V(I,K) + Y*V(I,K+1)                                     QRS  89Ø
          IF (LAST) GO TO 11Ø                                           QRS  9ØØ
          P = P + Z*V(I,K+2)                                            QRS  91Ø
          V(I,K+2) = V(I,K+2) - P*R                                     QRS  92Ø
  11Ø     CONTINUE                                                      QRS  93Ø
          V(I,K+1) = V(I,K+1) - P*Q                                     QRS  94Ø
          V(I,K) = V(I,K) - P                                           QRS  95Ø
  12Ø   CONTINUE                                                        QRS  96Ø
  13Ø CONTINUE                                                          QRS  97Ø
      RETURN                                                            QRS  98Ø
      END                                                               QRS  99Ø
```

ACM Transactions on Mathematical Software, Vol. 8, No. 2. June 1982, Pages 219-220.

REMARK ON ALGORITHM 506

HQR3 and EXCHNG: Fortran Subroutines for Calculating and Ordering the Eigenvalues of a Real Upper Hessenberg Matrix [G.W. Stewart, *ACM Trans. Math. Softw. 2*, 3(Sept. 1976), 275-280]

David S. Flamm and Robert A. Walker [Received 21 September 1981, accepted 20 February 1982]

D.S. Flamm, Aerospace Corporation, P.O. Box 92957, Los Angeles, CA 90009; R.A. Walker, Integrated Systems, 151 University Avenue, Suite 400, Palo Alto, CA 94301.

The following four corrections should be made to program HQR3 [1, 2]:

(1) Line HQR 720 in [2] should have the comparison ".LE." instead of ".LT.".
(2) Line HQR 1850 in [2] should branch to "220" instead of "230".
(3) Line HQR 2130 should read "DO 250 I = NLOW, NU", instead of "DO 250 I = 1, NU".
(4) Line EXC 410 should read "R = AMAX1(ABS(P), ABS(Q))" instead of "R = AMAX1(P,Q)".

The following example illustrates why the first two corrections are necessary. Apply HQR3 to the matrix

$$\begin{bmatrix} 0 & -2 & 0 & 0 & 0 & 0 \\ 3 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}.$$

An uncorrected version of HQR3 will yield a division by zero in line HQR 1060 of [2] as a result of the test in (1). In checking for a zero subdiagonal element, the criterion is the sum of the magnitudes of the two diagonals. As the example shows, when both diagonals are zero, the .LT. test fails for an exactly zero subdiagonal element.

Without correction (2), when exchanging two upper real roots with a lower 2 × 2 block, only the bottom root is tested and exchanged, leaving the possibility of misordering of the top root. The example above also demonstrates this problem.

Without correction (3), the program would give the wrong diagonals in an unusual case where part of a matrix was being triangularized and the routine failed to converge.

Correction (4) is necessary when the superdiagonal element is zero between two real roots, leaving the possibility that the maximum of (P,Q) is P = 0 for Q negative. (If (P,Q) are both zero, no exchange is required and the program properly returns.)

REFERENCES

1. STEWART, G.W. HQR3 and EXCHNG: Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix. *ACM Trans. Math. Softw. 2*, 3 (Sept. 1976), 275-280.
2. ALGORITHM 506. *Collected Algorithms from ACM*, ACM, New York.

ALGORITHM 507
Procedures for Quintic Natural Spline Interpolation [E1]

JOHN G. HERRIOT
Stanford University
and
CHRISTIAN H. REINSCH
Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften, Germany

Key Words and Phrases: approximation, interpolation, spline, spline approximation, quintic natural spline
CR Categories: 5.13
Language: Algol W

DESCRIPTION

1. Introduction

The purpose of the procedures presented here is to determine the interpolating quintic natural spline function $S(x)$ for the set of data points (x_i,y_i), $i = N1, N1+1, \ldots, N2$, where it is assumed that $x_{N1} < x_{N1+1} < \cdots < x_{N2}$. The interpolating quintic natural spline function $S(x)$ with the knots $x_{N1}, \ldots, x_{N2}$ has the following properties: (i) $S(x)$ is a polynomial of degree 5 in each interval (x_i, x_{i+1}), $i = N1, \ldots, N2-1$. (ii) $S(x)$ and its derivatives $S'(x)$, $S''(x)$, $S'''(x)$, and $S''''(x)$ are continuous in $[x_{N1},x_{N2}]$. (iii) $S'''(x_{N1}) = S'''(x_{N2}) = S''''(x_{N1}) = S''''(x_{N2}) = 0$. (iv) $S(x_i) = y_i$, $i = N1, \ldots, N_2$. It is known that if $N2 > N1+1$, then there is a unique quintic natural spline function which has the properties (i)–(iv). (See, for example, Greville [3, 4].) This spline function can be represented in the form

$$S(x) = y_i + B_i t + C_i t^2 + D_i t^3 + E_i t^4 + F_i t^5 \qquad (1)$$

with $t = x - x_i$ for $x_i \leq x < x_{i+1}$, $i = N1, \ldots, N2-1$.

The procedure *QUINAT* computes the coefficients B_i, C_i, D_i, E_i, F_i of the quintic natural spline represented as in eq. (1) for an arbitrary set of data points (x_i,y_i) as previously specified. This procedure is much faster than the procedure *NATSPLINE* of ACM Algorithm 472 [6] with $m = 3$. An even faster procedure, *QUINEQ*, is provided for the case in which the knots x_i are known to be equidistant. In this case it is not necessary to specify the values of x_i. The representation (1) is still used, but now $t = (x-x_i)/h$, where $h = x_{i+1}-x_i$, the constant spacing of the knots.

Received 20 June 1974.

This work was supported in part by the National Science Foundation, under Grant GJ-29988X.
Authors' addresses: J.G Herriot, Department of Computer Science, Stanford University Stanford, CA 94305; C.H. Reinsch, Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften, 8 München 2, Germany.

If at one or more of the knots x_i one also specifies the derivative y_i', thus requiring $S'(x_i) = y_i'$, then one has to give up the condition that $S''''(x)$ be continuous at the knot x_i. If the second derivative y_i'' is also specified, thus requiring $S''(x_i) = y_i''$, then one must also give up the condition that $S'''(x)$ be continuous at x_i. *QUINAT* is designed so that it can be used in these cases with the convention that if two consecutive knots are equal, say $x_j = x_{j+1}$, then $S(x_j) = y_j$ and $S'(x_j) = y_{j+1}$, and if three consecutive knots are equal, say $x_j = x_{j+1} = x_{j+2}$, then $S(x_j) = y_j$, $S'(x_j) = y_j$, and $S''(x_j) = y_{j+2}$. Thus in order to use *QUINAT* in the case that both the value y_j and the first derivative y_j' are specified at x_j, one increases the number of knots by 1 setting $x_{j+1} = x_j$ (and renumbering the knots and values to the right). Then one chooses $y_{j+1} = y_j'$. The spline function computed by *QUINAT* will have the property $S(x_j) = y_j$ and $S'(x_j) = y_{j+1}$. One may use *QUINAT* in a similar manner if the second derivative is also specified at a knot x_j. Complete details are given in the comment of the procedure *QUINAT*.

If the values of the function y_i and the values of the first derivative y_i' are specified at all the knots x_i, then $S''''(x)$ need not be continuous at the knots and $S''''(x_{N1})$ and $S''''(x_{N2})$ need not be zero. Such a spline is said to be of deficiency 2. The procedure *QUINDF* computes the coefficients of the quintic natural spline of deficiency 2 when the values of the function y_i and the values of the first derivative y_i' are given at each knot. Although *QUINAT* could be used for this case as just described, *QUINDF* is much faster and needs much less storage space.

It is not of interest to specify the values of the function and its first and second derivatives at each knot, because in this case the quintic polynomial is completely determined in each interval independently of all other intervals.

2. Method of Calculation

QUINAT. As in the general case of Algorithm 472 [6], the calculation of the coefficients of the spline function is carried out in a numerically stable manner following a method described by Anselone and Laurent [1]. The basic ideas on which the method is based were given earlier by Schoenberg [7]. The method is specialized to the case of the quintic natural spline and uses minimum support B-splines [2, 3] of degree 2 to form a basis for the class of third derivatives of the quintic natural splines. Instead of specializing the formulas of Algorithm 472 [6] by setting $m = 3$, we derive the necessary formulas directly and we choose a different numbering and a different normalization for the B-splines.

We first assume that the knots are strictly monotone increasing, that is, $x_{N1} < x_{N1+1} < \cdots < x_{N2}$. In order to simplify the notation, we choose $N1 = 0$ and let $N2 = n$, so that the data points are denoted by (x_i, y_i), $i = 0, 1, \ldots, n$. We denote by $M_i(x)$ the B-spline of degree 2, vanishing outside the interval (x_{i-1}, x_{i+2}). We let $h_i = x_{i+1} - x_i$, $t = x - x_{i-1}$, $u = x - x_i$, $v = x - x_{i+1}$.

Then we have

$$\begin{aligned} M_i(x) &= At^2, && x_{i-1} \le x < x_i, \\ &= B + Cu - Du^2, && x_i \le x < x_{i+1}, \\ &= E(v - h_{i+1})^2, && x_{i+1} \le x < x_{i+2}, \end{aligned} \tag{2}$$

where

$$A = 1/[h_{i-1}(h_{i-1} + h_i)], \quad B = h_{i-1}/(h_{i-1} + h_i), \quad C = 2/(h_{i-1} + h_i), \tag{3}$$

$$D = (h_{i-1} + 2h_i + h_{i+1})/[(h_{i-1} + h_i)h_i(h_i + h_{i+1})], \quad E = 1/[h_{i+1}(h_i + h_{i+1})].$$

Now since the third derivative $S'''(x)$ vanishes outside the interval (x_0, x_n), it has a unique representation of the form

$$S'''(x) = \sum_{j=1}^{n-2} 60\gamma_j M_j(x). \tag{4}$$

In order to determine the γ_j, we make use of the relation

$$\int_{-\infty}^{\infty} M_i(x) S'''(x) dx = 2(S(x_i, x_{i+1}, x_{i+2}) - S(x_{i-1}, x_i, x_{i+1})) \tag{5}$$

using the usual notation for divided differences. This relation is easily obtained by integration by parts. If we multiply eq. (4) by $\frac{1}{2}M_i(x)$, $i = 1, 2, \ldots, n-2$, and integrate, we obtain a well-conditioned positive definite pentadiagonal system of linear equations for the determination of the γ_j:

$$\begin{aligned}
d_1\gamma_1 + e_1\gamma_2 + f_1\gamma_3 &= c_1 \\
e_1\gamma_1 + d_2\gamma_2 + e_2\gamma_3 + f_2\gamma_4 &= c_2 \\
f_{i-2}\gamma_{i-2} + e_{i-1}\gamma_{i-1} + d_i\gamma_i + e_i\gamma_{i+1} + f_i\gamma_{i+2} &= c_i, \quad i = 3, 4, \ldots, n-4 \qquad (6) \\
f_{n-5}\gamma_{n-5} + e_{n-4}\gamma_{n-4} + d_{n-3}\gamma_{n-3} + e_{n-3}\gamma_{n-2} &= c_{n-3} \\
f_{n-4}\gamma_{n-4} + e_{n-3}\gamma_{n-3} + d_{n-2}\gamma_{n-2} &= c_{n-2}
\end{aligned}$$

where

$$\begin{aligned}
d_i &= T_1 + T_2 + T_3, & i &= 1,2,\ldots,n-2, \\
e_i &= T_4 + T_5, & i &= 1,2,\ldots,n-3, \\
f_i &= T_6, & i &= 1,2,\ldots,n-4, \\
c_i &= y_{i,i+1,i+2} - y_{i-1,i,i+1}, & i &= 1,2,\ldots,n-2.
\end{aligned}$$

Here $y_{i,i+1,i+2}$ denotes the second divided difference of the given $\{y_i\}$, and for the T_i one finds, after some algebraic manipulation, the following formulas:

$$\begin{aligned}
T_1 &= 6h_{i-1}^3/(h_{i-1} + h_i)^2 \\
T_2 &= h_i\{30h_{i-1}^2h_{i+1}^2 + (h_{i-1} + h_{i+1})h_i(40h_{i-1}h_{i+1} + 14h_i^2) \\
&\quad + h_i^2[16(h_{i-1}^2 + h_{i+1}^2) + 42h_{i-1}h_{i+1} + 4h_i^2]\}/[(h_{i-1} + h_i)^2(h_i + h_{i+1})^2] \\
T_3 &= 6h_{i+1}^3/(h_i + h_{i+1})^2 \\
T_4 &= h_i^2[h_{i-1}(h_i + h_{i+1}) + 3(h_{i-1} + h_i)(h_i + 3h_{i+1})]/[(h_{i-1} + h_i)(h_i + h_{i+1})^2] \\
T_5 &= h_{i+1}^2[h_{i+2}(h_i + h_{i+1}) + 3(h_{i+1} + h_{i+2})(3h_i + h_{i+1})]/[(h_i + h_{i+1})^2(h_{i+1} + h_{i+2})] \\
T_6 &= h_{i+1}^3/[(h_i + h_{i+1})(h_{i+1} + h_{i+2})].
\end{aligned}$$

Note that all terms in these expressions are positive; consequently no cancellations can occur. The system of equations (6) can be solved for the γ_j by using Gaussian elimination without pivoting. When the coefficients γ_j have been found, $S'''(x)$ is given by eq. (4). Remembering that $M_j(x)$ vanishes outside the interval (x_{j-1}, x_{j+2}) and making use of eqs. (1) through (4), we easily find that

$$\left.\begin{aligned}
D_i/10 &= (\gamma_{i-1}h_i + \gamma_i h_{i-1})/(h_{i-1} + h_i) \\
E_i/5 &= (\gamma_i - \gamma_{i-1})/(h_{i-1} + h_i) \\
F_i &= (1/h_i)[(\gamma_{i+1} - \gamma_i)/(h_i + h_{i+1}) \\
&\quad - (\gamma_i - \gamma_{i-1})/(h_{i-1} + h_i)].
\end{aligned}\right\} \quad i = 2, 3, \ldots, n-3.$$

These formulas can also be used for $i = 0, 1, n-2, n-1$ by adding the convention that $\gamma_{-1} = \gamma_0 = \gamma_{n-1} = \gamma_n = 0$. (Note that $D_0 = E_0 = 0$ as they should.) Finally we make use of the continuity of $S(x)$ and its first four derivatives at x_i to obtain the following formulas for B_i and C_i:

$$\begin{aligned}
B_i &= \frac{h_{i-1}}{h_{i-1} + h_i}\frac{y_{i+1} - y_i}{h_i} + \frac{h_i}{h_{i-1} + h_i}\frac{y_i - y_{i-1}}{h_{i-1}} - D_ih_{i-1}h_i + E_ih_{i-1}h_i(h_{i-1} - h_i) \\
&\quad - \frac{h_{i-1}h_i}{h_{i-1} + h_i}(F_{i-1}h_{i-1}^3 + F_ih_i^3) \\
C_i &= \frac{1}{h_{i-1} + h_i}\left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right) + D_i(h_{i-1} - h_i) \\
&\quad - E_i\frac{h_{i-1}^3 + h_i^3}{h_{i-1} + h_i} + \frac{1}{h_{i-1} + h_i}(F_{i-1}h_{i-1}^4 - F_ih_i^4).
\end{aligned}$$

These formulas are valid for $i = 1, 2, \ldots, n-1$. In addition, we have for the endpoints:

$$C_0 = C_1 - 10F_0h_0^3, \qquad B_0 = (y_1 - y_0)/h_0 - C_0h_0 - F_0h_0^4,$$

$$C_n = C_{n-1} + 10F_{n-1}h_{n-1}^3, \qquad B_n = (y_n - y_{n-1})/h_{n-1} + C_n h_{n-1} - F_{n-1}h_{n-1}^4.$$

In the preceding discussion we have assumed that the knots were distinct. We can relax this condition and allow two or three consecutive knots to be equal. The procedure *QUINAT* has been written in such a way that if $x_j = x_{j+1}$, then $S(x_j) = y_j$ and $S'(x_j) = y_{j+1}$, and if $x_j = x_{j+1} = x_{j+2}$, then, in addition, $S''(x_j) = y_{j+2}$. The use of *QUINAT* in these cases is fully explained in its comment.

QUINEQ. The calculation of the coefficients in *QUINEQ* for the case of equidistant knots is carried out in the same manner as is the calculation of the coefficients in *QUINAT* for the general case. However, there are a number of simplifications which result in considerable economy of computational effort. It is not necessary to specify x_i. Hence we can assume $x_i = i$. Then $h_i = 1$ for all i, and the coefficients of $M_i(x)$ are independent of i as are also the d_i, e_i, f_i of the pentadiagonal system (6) for the γ_i. Thus eqs. (2) reduce to

$$M_i(x) = \begin{cases} \frac{1}{2}t^2, & i-1 \le x < i, \\ \frac{1}{2} + u - u^2, & i \le x < i+1, \\ \frac{1}{2}(v-1)^2, & i+1 \le x < i+2, \end{cases}$$

with $t = x - (i-1)$, $u = x - i$, $v = x - (i+1)$.

Instead of eq. (4) it is convenient to take

$$S'''(x) = \sum_{j=0}^{n-3} 120\gamma_j M_{j+1}(x).$$

The divided differences become ordinary differences so that eq. (5) becomes:

$$\int_{-\infty}^{\infty} M_i(x)S'''(x)\,dx = \Delta^3 S(x_{i-1}).$$

The pentadiagonal system (6) for the determination of γ_j becomes:

$$\begin{aligned}
66\gamma_0 + 26\gamma_1 + \gamma_2 &= \Delta^3 y_0 \\
26\gamma_0 + 66\gamma_1 + 26\gamma_2 + \gamma_3 &= \Delta^3 y_1 \\
\gamma_{i-2} + 26\gamma_{i-1} + 66\gamma_i + 26\gamma_{i+1} + \gamma_{i+2} &= \Delta^3 y_i, \quad i = 2, 3, \ldots, n-5 \\
\gamma_{n-6} + 26\gamma_{n-5} + 66\gamma_{n-4} + 26\gamma_{n-3} &= \Delta^3 y_{n-4} \\
\gamma_{n-5} + 26\gamma_{n-4} + 66\gamma_{n-3} &= \Delta^3 y_{n-3}.
\end{aligned}$$

The equations for the determination of the spline function coefficients then become:

$$\begin{aligned}
D_i/10 &= \gamma_{i-2} + \gamma_{i-1} & B_i &= \tfrac{1}{2}(y_{i+1} - y_{i-1} - F_{i-1} - F_i) - D_i \\
E_i/5 &= \gamma_{i-1} - \gamma_{i-2} & C_i &= \tfrac{1}{2}(y_{i+1} + y_{i-1} + F_{i-1} - F_i) - y_i - E_i. \\
F_i &= \gamma_i - \gamma_{i-1} - \gamma_{i-1} + \gamma_{i-2}
\end{aligned}$$

These formulas are valid for $i = 1, 2, \ldots, n-1$ with the convention that $\gamma_{-1} = \gamma_{n-2} = \gamma_{n-1} = 0$. The formula for F_i can be used for $i = 0$ by setting $\gamma_{-2} = 0$. (Note that $D_0 = E_0 = 0$ as they should.) Finally the coefficients B_i and C_i at the endpoints are given by

$$\begin{aligned}
C_0 &= C_1 - 10F_0, & B_0 &= y_1 - y_0 - C_0 - F_0, \\
C_n &= C_{n+1} + 10F_{n-1}, & B_n &= y_n - y_{n-1} + C_n - F_{n-1}.
\end{aligned}$$

QUINDF. We now assume $S(x_i) = y_i$ and $S'(x_i) = y_i'$ are specified at each of the knots. We must exclude the possibility that $x_i = x_{i+1}$ as this would imply a multiplicity of 4, which is not feasible for quintic splines.

We could proceed as in the calculation of *QUINAT* by using minimum support B-splines of degree 2 to form a basis for the class of third derivatives of the quintic natural splines. Of course, the B-splines would also have to be of deficiency 2. We

would again obtain a pentadiagonal system of equations which could be solved and then the coefficients for the deficient quintic natural spline could be calculated. An algorithm based on this method was developed and tested by the present authors [5].

However, we have found that we can obtain a more efficient algorithm by imposing the appropriate continuity conditions directly on eq. (1) at the knots. A similar method was used by Späth [8] to obtain an algorithm for the deficient quintic spline but with different end conditions. (He specified the second derivatives at the endpoints of the interval instead of requiring the third derivative to be zero at the endpoints as for the natural spline.)

Using eq. (1) we see at once that $S'(x_i) = y_i'$ implies $B_i = y_i'$. Then imposing the requirement that $S(x)$, $S'(x)$, and $S''(x)$ be continuous at x_{i+1}, $i = 0, 1, \ldots, n-2$, we obtain by setting $t = h_i = x_{i+1} - x_i$ in eq. (1):

$$\begin{aligned} y_{i+1} &= y_i + B_i h_i + C_i h_i^2 + D_i h_i^3 + E_i h_i^4 + F_i h_i^5 \\ B_{i+1} &= B_i + 2C_i h_i + 3D_i h_i^2 + 4E_i h_i^3 + 5F_i h_i^4 \\ C_{i+1} &= C_i + 3D_i h_i + 6E_i h_i^2 + 10F_i h_i^3. \end{aligned} \tag{7}$$

If we multiply these three equations by $10/h_i^3$, $-4/h_i^2$, $1/h_i$, respectively, and add the results, we eliminate E_i and F_i and obtain

$$D_i = 10\,\frac{y_{i+1} - y_i}{h_i^3} - \frac{4B_{i+1} + 6B_i}{h_i^2} + \frac{C_{i+1} - 3C_i}{h_i}. \tag{8}$$

We note that $D_i = S'''(x_i + 0)/6$. In order to obtain a similar expression for $S'''(x_i - 0)/6$, we replace the subscript $i+1$ by $i-1$ consistently (noting that then $h_i = x_{i+1} - x_i$ is to be replaced by $x_{i-1} - x_i = -h_{i-1}$). We obtain

$$\tfrac{1}{6}S'''(x_i - 0) = 10\,\frac{y_i - y_{i-1}}{h_{i-1}^3} - \frac{6B_i + 4B_{i-1}}{h_{i-1}^2} + \frac{3C_i - C_{i-1}}{h_{i-1}}. \tag{9}$$

Now since the third derivative is continuous at x_i, we can equate the values of $S'''(x_i + 0)/6$ and $S'''(x_i - 0)/6$ in eqs. (8) and (9) to obtain the following set of equations for the C_i:

$$\begin{aligned} -\frac{1}{h_{i-1}} C_{i-1} + \left(\frac{3}{h_{i-1}} + \frac{3}{h_i}\right) C_i - \frac{1}{h_i} C_{i+1} \\ = 10\left(\frac{y_{i+1} - y_i}{h_i^3} - \frac{y_i - y_{i-1}}{h_{i-1}^3}\right) - \frac{4B_{i+1} + 6B_i}{h_i^2} + \frac{6B_i + 4B_{i-1}}{h_{i-1}^2}. \end{aligned}$$

These equations hold for $i = 1, 2, \ldots, n-1$. Two additional equations can be obtained from the conditions $S'''(x_0) = S'''(x_n) = 0$ by setting $i = 0$ in eq. (8) and by setting $i = n$ in eq. (9):

$$\begin{aligned} \frac{3}{h_0} C_0 - \frac{1}{h_0} C_1 &= 10\,\frac{y_1 - y_0}{h_0^3} - \frac{4B_1 + 6B_0}{h_0^2}, \\ -\frac{1}{h_{n-1}} C_{n-1} + \frac{3}{h_n} C_n &= -10\,\frac{y_n - y_{n-1}}{h_{n-1}^3} + \frac{6B_n + 4B_{n-1}}{h_{n-1}^2}. \end{aligned}$$

This system of $n+1$ equations is solved to obtain the C_i. If we make the substitutions

$$\begin{aligned} x &= D_i, & p &= (y_{i+1} - y_i - B_i h_i - C_i h_i^2)/h_i^3, \\ y &= E_i h_i, & q &= (B_{i+1} - B_i - 2C_i h_i)/h_i^2, \\ z &= F_i h_i^2, & r &= (C_{i+1} - C_i)/h_i, \end{aligned}$$

then for each i eqs. (7) form a system of three equations in the three unknowns x, y, z, and the system is solved by Gaussian elimination. The backward substitution

yields formulas for z, y, x in the following order:

$$\begin{cases} g &= q - 3p \\ z &= r - 3(p + g) \\ y &= g - z - z \\ x &= p - y - z \\ F_i &= z/h_i^2 \\ E_i &= y/h_i \\ D_i &= x. \end{cases}$$

3. Tests

These procedures have been tested in Algol 60 on the Telefunken TR-440 computer at the Leibniz-Rechenzentrum of the Bavarian Academy of Sciences, Munich, and in Algol W on the IBM 360/67 at the Stanford Center for Information Processing. The latter tests included timing tests of the procedures with the number of knots $N = N2 - N1 + 1$ ranging up to 1000. The time was found to be approximately proportional to the number N of knots. The time T in seconds for the execution of the procedure *QUINAT* was found to be approximately $T = .00193N$, whereas for the procedure *NATSPLINE* of Algorithm 472 [6] with $m = 3$ it was found to be $T = .0120N$, or over six times as great. For the procedure *QUINEQ* the time was approximately $T = .00064N$, whereas for the procedure *NATSPLINEEQ* of Algorithm 472 [6] with $m = 3$ it was $T = .0038N$, or nearly six times as great. For the procedure *QUINDF* the time was approximately $T = .00087N$, whereas for the procedure *QUINAT* with $2N$ knots, consecutive knots being equal in pairs, the time was $T = .00325N$, or nearly four times as great. Moreover, to compute the same results the procedure *QUINAT* requires approximately twice as much storage for the arrays used as does the procedure *QUINDF*. Note also that from the preceding formula for the time required by the procedure *QUINAT*, the time for $2N$ distinct knots would be $T = .00386N$, which can be compared with $T = .00325N$ given above for N pairs of equal knots. The reduction for the case of double knots occurs because some calculations are omitted when knots are coincident.

These timing comparisons show that it is definitely advantageous to use these special procedures for the quintic natural spline rather than the general cases given in Algorithm 472 [6] with $m = 3$.

Tests of the accuracy and correctness of the coefficients computed by the procedures *QUINAT*, *QUINEQ*, and *QUINDF* were carried out as described in Algogorithm 472 [6]. Table I shows the results of a typical run using *QUINDF* for 5 nonequidistant points. The values of the function and its first derivatives were specified. The first line of each entry gives the tabulated quantities at the given

Table I. Quintic Spline Calculated by *QUINDF*
(Machine precision approximately 7 decimal digits.)

x	S(x)	S'(x)	S"(x)/2	S"'(x)/6	S""(x)/24	$S^V(x)/120$
-3.000000	7.000000	2.000000	-6.108377	-5.722046'-06	2.956286	-0.7145951
	11.00000	15.00000	7.674876	-4.933508	-4.189662	-0.7145951
-1.000000	11.00000	15.00000	7.674870	-4.933474	-8.157658	5.416262
	26.00000	9.999996	-1.908880	16.59851	18.92365	5.416262
0	26.00000	10.00000	-1.908880	16.59848	-9.059000	1.246088
	55.99942	-27.00066	-5.264791	20.03839	9.632320	1.246088
3.000000	56.00000	-27.00000	-5.264426	20.03847	-21.28366	6.509618
	29.00000	-30.00000	-7.754811	-1.907349'-06	11.26443	6.509618
4.000000	29.00000	-30.00000				

value of x which is the lefthand endpoint of the subinterval; the second line of each entry gives the tabulated values at the righthand endpoint of the same subinterval. The close agreement of the quantities $S(x)$, $S'(x)$, $S''(x)/2$, and $S'''(x)/6$ shows that the quintic spline function and its derivatives satisfy the required continuity conditions. This is a good indication of the correctness of the results. Note that the fourth and fifth derivatives are discontinuous. Essentially the same results were obtained by using *QUINAT* with 10 knots, in equal pairs. In addition, accuracy and timing tests were carried out for large values of N, including $N = 1000$ and 5000, and produced very satisfactory results.

REFERENCES

1. ANSELONE, P.M., AND LAURENT, P.J. A general method for the construction of interpolating and smoothing spline functions. *Numer. Math. 12* (1968), 66–82.
2. CURRY, H.B., AND SCHOENBERG, I.J. On Pólya frequency functions. IV. The fundamental spline functions and their limits. *J. Anal. Math. 17* (1966), 71–107.
3. GREVILLE, T.N.E. Introduction to spline functions. In *Theory and Applications of Spline Functions*, T.N.E. Greville, Ed. Academic Press, New York, 1969, pp. 1–35.
4. GREVILLE, T.N.E. Spline functions, interpolation and numerical quadrature. In *Mathematical Methods for Digital Computers, vol. II*, A. Ralston and H.S. Wilf, Eds. Wiley, New York, 1967.
5. HERRIOT, J.G., AND REINSCH, C.H. Algol 60 procedures for the calculation of interpolating natural quintic spline functions. Tech. Rep. STAN-CS-74-402, Dep. Computer Science, Stanford Univ., Stanford, Calif., 1974.
6. HERRIOT, J.G., AND REINSCH, C.H. Algorithm 472, Procedures for natural spline interpolation. *Comm. ACM 16*, 12 (Dec. 1973), 763–768.
7. SCHOENBERG, I.J. On interpolation by spline functions and its minimal properties. In *On Approximation Theory. Proceedings of the Conference at Oberwolfach, 1963*, P.L. Butzer and J. Korevaar, Eds. Birkhaüser Verlag, Basel, Switzerland, 1964, pp. 109–129.
8. SPÄTH, H. Algorithm 42, Interpolation by certain quintic splines. *Computer J. 12* (1969), 292–293.

ALGORITHM

```
PROCEDURE QUINAT(INTEGER VALUE N1,N2; REAL ARRAY X,Y,B,C,D,E,F(*));
COMMENT  QUINAT COMPUTES THE COEFFICIENTS OF A QUINTIC NATURAL SPLINE
   S(X) INTERPOLATING THE ORDINATES Y(I) AT POINTS X(I), I = N1
   THROUGH N2.  FOR XX IN (X(I),X(I+1)) THE VALUE OF THE SPLINE
   FUNCTION S(XX) IS GIVEN BY THE FIFTH DEGREE POLYNOMIAL:
   S(XX) = ((((F(I)*T+E(I))*T+D(I))*T+C(I))*T+B(I))*T+Y(I)
   WITH T = XX - X(I).
   INPUT:
     N1,N2  SUBSCRIPT OF FIRST AND LAST DATA POINT RESPECTIVELY,
            IT IS REQUIRED THAT N2 > N1 + 1,
     X,Y(N1::N2)  ARRAYS WITH X(I) AS ABSCISSA AND Y(I) AS ORDINATE
            OF THE I-TH DATA POINT.  THE ELEMENTS OF THE ARRAY X
            MUST BE STRICTLY MONOTONE INCREASING (BUT SEE BELOW FOR
            EXCEPTIONS TO THIS).
   OUTPUT:
     B,C,D,E,F(N1::N2)  ARRAYS COLLECTING THE COEFFICIENTS OF THE
            QUINTIC NATURAL SPLINE S(XX) AS DESCRIBED ABOVE.
            SPECIFICALLY  B(I) = S'(X(I)),  C(I) = S"(X(I))/2,
            D(I) = S"'(X(I))/6,  E(I) = S""(X(I))/24,
            F(I) = S""'(X(I)+0)/120.  F(N2) IS NEITHER USED OR
            ALTERED.  THE ARRAYS B,C,D,E,F MUST ALWAYS BE DISTINCT.
   OPTIONS:
     1.  THE REQUIREMENT THAT THE ELEMENTS OF THE ARRAY X BE
         STRICTLY MONOTONE INCREASING CAN BE RELAXED TO ALLOW TWO
         OR THREE CONSECUTIVE ABSCISSAS TO BE EQUAL AND THEN
         SPECIFYING VALUES OF THE FIRST AND SECOND DERIVATIVES OF
         THE SPLINE FUNCTION AT SOME OF THE INTERPOLATING POINTS.
         SPECIFICALLY
         IF X(J) = X(J+1) THEN S(X(J)) = Y(J) AND S'(X(J)) = Y(J+1),
         IF X(J) = X(J+1) = X(J+2) THEN IN ADDITION S"(X(J)) =Y(J+2).
         NOTE THAT S""(X) IS DISCONTINUOUS AT A DOUBLE KNOT AND IN
         ADDITION S"'(X) IS DISCONTINUOUS AT A TRIPLE KNOT.  AT A
         DOUBLE KNOT, X(J) = X(J+1), THE OUTPUT COEFFICIENTS HAVE THE
         FOLLOWING VALUES:
           B(J) = S'(X(J))       = B(J+1)
           C(J) = S"(X(J))/2     = C(J+1)
           D(J) = S"'(X(J))/6    = D(J+1)
           E(J) = S""(X(J)-0)/24     E(J+1) = S""(X(J)+0)/24
           F(J) = S""'(X(J)-0)/120  F(J+1) = S""'(X(J)+0)/120
         THE REPRESENTATION OF S(XX) REMAINS VALID IN ALL INTERVALS
         PROVIDED THE REDEFINITION Y(J+1) := Y(J) IS MADE
```

```
          IMMEDIATELY AFTER THE CALL OF THE PROCEDURE QUINAT.  AT A      42.
          TRIPLE KNOT, X(J) = X(J+1) = X(J+2), THE OUTPUT COEFFICIENTS   43.
          HAVE THE FOLLOWING VALUES:                                     44.
            B(J) = S'(X(J))        = B(J+1) = B(J+2)                     45.
            C(J) = S"(X(J))/2      = C(J+1) = C(J+2)                     46.
            D(J) = S"'(X(J)-0)/6   D(J+1) = 0  D(J+2) = S"'(X(J)+0)/6    47.
            E(J) = S""(X(J)-0)/24  E(J+1) = 0  E(J+2) = S""(X(J)+0)/24   48.
            F(J) = S""'(X(J)-0)/120  F(J+1)=0  F(J+2)=S""'(X(J)+0)/120   49.
          THE REPRESENTATION OF S(XX) REMAINS VALID IN ALL INTERVALS     50.
          PROVIDED THE REDEFINITION Y(J+2) := Y(J+1) := Y(J) IS MADE     51.
          IMMEDIATELY AFTER THE CALL OF THE PROCEDURE QUINAT.            52.
        2. THE ARRAY X MAY BE MONOTONE DECREASING INSTEAD OF             53.
          INCREASING;                                                    54.
IF N2 > N1 + 1 THEN                                                      55.
BEGIN                                                                    56.
  INTEGER M;                                                             57.
  REAL B1,P,PQ,PQQR,PR,P2,P3,Q,QR,Q2,Q3,R,R2,S,T,U,V;                    58.
  COMMENT COEFFICIENTS OF A POSITIVE DEFINITE, PENTADIAGONAL             59.
     MATRIX STORED IN D,E,F(N1+1::N2-2);                                 60.
  M:=N2-2;                                                               61.
  Q:=X(N1+1)-X(N1);  R:=X(N1+2)-X(N1+1);                                 62.
  Q2:=Q*Q;  R2:=R*R;  QR:=Q+R;                                           63.
  D(N1):=E(N1):=0.0;                                                     64.
  D(N1+1):=IF Q=0.0 THEN 0.0 ELSE 6.0*Q*Q2/(QR*QR);                      65.
  FOR I:=N1+1 STEP 1 UNTIL M DO                                          66.
  BEGIN                                                                  67.
     P:=Q;  Q:=R;  R:=X(I+2)-X(I+1);                                     68.
     P2:=Q2;  Q2:=R2;  R2:=R*R;  PQ:=QR;  QR:=Q+R;                       69.
     IF Q=0.0 THEN D(I+1):=E(I):=F(I-1):=0.0 ELSE                        70.
     BEGIN                                                               71.
        Q3:=Q2*Q;  PR:=P*R;  PQQR:=PQ*QR;                                72.
        D(I+1):=6.0*Q3/(QR*QR);                                          73.
        D(I):=D(I)+(Q+Q)*(15.0*PR*PR+(P+R)*Q*(20.0*PR+7.0*Q2)            74.
                  +Q2*(8.0*(P2+R2)+21.0*PR+Q2+Q2))/(PQQR*PQQR);          75.
        D(I-1):=D(I-1)+6.0*Q3/(PQ*PQ);                                   76.
        E(I):=Q2*(P*QR+3.0*PQ*(QR+R+R))/(PQQR*QR);                       77.
        E(I-1):=E(I-1)+Q2*(R*PQ+3.0*QR*(PQ+P+P))/(PQQR*PQ);              78.
        F(I-1):=Q3/PQQR;                                                 79.
     END;                                                                80.
  END;                                                                   81.
  IF R=0.0 THEN D(M):=D(M)+6.0*R*R2/(QR*QR);                             82.
  COMMENT FIRST AND SECOND ORDER DIVIDED DIFFERENCES OF THE GIVEN        83.
     FUNCTION VALUES,STORED IN B(N1+1::N2) AND C(N1+2::N2)               84.
     RESPECTIVELY, TAKE CARE OF DOUBLE AND TRIPLE KNOTS;                 85.
  S:=Y(N1);                                                              86.
  FOR I:=N1+1 STEP 1 UNTIL N2 DO                                         87.
  IF X(I)=X(I-1) THEN B(I):=Y(I) ELSE                                    88.
  BEGIN                                                                  89.
     B(I):=(Y(I)-S)/(X(I)-X(I-1));                                       90.
     S:=Y(I);                                                            91.
  END;                                                                   92.
  FOR I:=N1+2 STEP 1 UNTIL N2 DO                                         93.
  IF X(I)=X(I-2) THEN                                                    94.
  BEGIN C(I):=Y(I)*0.5; B(I):=B(I-1) END                                 95.
  ELSE C(I):=(B(I)-B(I-1))/(X(I)-X(I-2));                                96.
  COMMENT SOLVE THE LINEAR SYSTEM WITH C(I+2)-C(I+1)                     97.
     AS RIGHT-HAND SIDE;                                                 98.
  IF M > N1 THEN                                                         99.
  BEGIN                                                                 100.
     P:=C(N1):=E(M):=F(N1):=F(M-1):=F(M):=0.0;                          101.
     C(N1+1):=C(N1+3)-C(N1+2); D(N1+1):=1.0/D(N1+1);                    102.
  END;                                                                  103.
  FOR I:=N1+2 STEP 1 UNTIL M DO                                         104.
  BEGIN                                                                 105.
     Q:=D(I-1)*E(I-1);                                                  106.
     D(I):=1.0/(D(I)-P*F(I-2)-Q*E(I-1));                                107.
     E(I):=E(I)-Q*F(I-1);                                               108.
     C(I):=C(I+2)-C(I+1)-P*C(I-2)-Q*C(I-1);                             109.
     P:=D(I-1)*F(I-1);                                                  110.
  END;                                                                  111.
  M:=N1+1;  C(N2-1):=C(N2):=0.0;                                        112.
  FOR I:=N2-2 STEP -1 UNTIL M DO                                        113.
     C(I):=(C(I)-E(I)*C(I+1)-F(I)*C(I+2))*D(I);                         114.
  COMMENT INTEGRATE THE THIRD DERIVATIVE OF S(X);                       115.
  M:=N2-1;                                                              116.
  Q:=X(N1+1)-X(N1); R:=X(N1+2)-X(N1+1); B1:=B(N1+1);                    117.
  Q3:=Q*Q*Q; QR:=Q+R;                                                   118.
  V:=T:=IF QR=0.0 THEN 0.0 ELSE C(N1+1)/QR;                             119.
  F(N1):=IF Q=0.0 THEN 0.0 ELSE V/Q;                                    120.
  FOR I:=N1+1 STEP 1 UNTIL M DO                                         121.
  BEGIN                                                                 122.
      P:=Q;  Q:=R;                                                      123.
      R:=IF I=N2-1 THEN 0.0 ELSE X(I+2)-X(I+1);                         124.
      P3:=Q3;  Q3:=Q*Q*Q; PQ:=QR;  QR:=Q+R;                             125.
```

```
      S:=T;  T:=IF QR=0.0 THEN 0.0 ELSE (C(I+1)-C(I))/QR;                126.
      U:=V; V:=T-S;                                                      127.
      IF PQ=0.0 THEN                                                     128.
      BEGIN  C(I):=0.5*Y(I+1); D(I):=E(I):=F(I):=0.0 END                 129.
      ELSE                                                               130.
      BEGIN                                                              131.
         F(I):=IF Q=0.0 THEN F(I-1) ELSE V/Q;                            132.
         E(I):=5.0*S;                                                    133.
         D(I):=10.0*(C(I)-Q*S);                                          134.
         C(I):=D(I)*(P-Q)+(B(I+1)-B(I)+(U-E(I))*P3                       135.
              -(V+E(I))*Q3)/PQ;                                          136.
         B(I):=(P*(B(I+1)-V*Q3)+Q*(B(I)-U*P3))/PQ                        137.
              -P*Q*(D(I)+E(I)*(Q-P));                                    138.
      END;                                                               139.
   END I;                                                                140.
   COMMENT END POINTS X(N1) AND X(N2);                                   141.
   P:=X(N1+1)-X(N1); S:=F(N1)*P*P*P;                                     142.
   E(N1):=D(N1):=0.0;                                                    143.
   C(N1):=C(N1+1)-10.0*S;                                                144.
   B(N1):=B1-(C(N1)+S)*P;                                                145.
   Q:=X(N2)-X(N2-1);  T:=F(N2-1)*Q*Q*Q;                                  146.
   E(N2):=D(N2):=0.0;                                                    147.
   C(N2):=C(N2-1)+10.0*T;                                                148.
   B(N2):=B(N2)+(C(N2)-T)*Q;                                             149.
END QUINAT;                                                              150.


PROCEDURE QUINEQ(INTEGER VALUE N1,N2; REAL ARRAY Y,B,C,D,E,F(*));        151.
COMMENT  QUINEQ COMPUTES THE COEFFICIENTS OF A QUINTIC NATURAL SPLINE    152.
   S(X) INTERPOLATING THE ORDINATES Y(I) AT EQUIDISTANT POINTS X(I),     153.
   I = N1 THROUGH N2.  FOR XX IN (X(I),X(I+1)) THE VALUE OF THE          154.
   SPLINE FUNCTION S(XX) IS GIVEN BY THE FIFTH DEGREE POLYNOMIAL:        155.
   S(XX) = ((((F(I)*T+E(I))*T+D(I))*T+C(I))*T+B(I))*T+Y(I)               156.
   WITH T = (XX - X(I))/(X(I+1) - X(I)).                                 157.
   INPUT:                                                                158.
     N1, N2  SUBSCRIPT OF FIRST AND LAST DATA POINT RESPECTIVELY,        159.
             IT IS REQUIRED THAT N2 > N1 + 1,                            160.
     Y(N1::N2)  THE GIVEN FUNCTION VALUES (ORDINATES).                   161.
   OUTPUT:                                                               162.
     B,C,D,E,F(N1::N2)  ARRAYS COLLECTING THE COEFFICIENTS OF THE        163.
             QUINTIC NATURAL SPLINE S(XX) AS DESCRIBED ABOVE.            164.
             SPECIFICALLY B(I) = S'(X(I)),  C(I) = S"(X(I))/2,           165.
             D(I) = S"'(X(I))/6,  E(I) = S""(X(I))/24,                   166.
             F(I) = S""'(X(I)+0)/120.  F(N2) IS NEITHER USED             167.
             NOR ALTERED.  THE ARRAYS Y,B,C,D MUST ALWAYS BE             168.
             DISTINCT.  IF E AND F ARE NOT WANTED, THE CALL              169.
             QUINEQ(N1,N2,Y,B,C,D,D,D) MAY BE USED TO SAVE STORAGE       170.
             LOCATIONS;                                                  171.
IF N2>N1+1 THEN                                                          172.
BEGIN                                                                    173.
   INTEGER N;                                                            174.
   REAL P,Q,R,S,T,U,V;                                                   175.
   N:=N2-3;  P:=Q:=R:=S:=T:=0.0;                                         176.
   FOR I:=N1 STEP 1 UNTIL N DO                                           177.
   BEGIN                                                                 178.
      U:=P*R;  B(I):=1.0/(66.0-U*R-Q);                                   179.
      C(I):=R:=26.0-U;                                                   180.
      D(I):=Y(I+3)-3.0*(Y(I+2)-Y(I+1))-Y(I)-U*S-Q*T;                     181.
      Q:=P;  P:=B(I);  T:=S;  S:=D(I)                                    182.
   END I;                                                                183.
   D(N+1):=D(N+2):=0.0;                                                  184.
   FOR I:=N STEP -1 UNTIL N1 DO                                          185.
      D(I):=(D(I)-C(I)*D(I+1)-D(I+2))*B(I);                              186.
   N:=N2-1; Q:=0.0;  R:=T:=V:=D(N1);                                     187.
   FOR I:=N1+1 STEP 1 UNTIL N DO                                         188.
   BEGIN                                                                 189.
      P:=Q;  Q:=R;  R:=D(I); S:=T;                                       190.
      F(I):=T:=P-Q-Q+R;                                                  191.
      E(I):=U:=5.0*(-P+Q);                                               192.
      D(I):=10.0*(P+Q);                                                  193.
      C(I):=0.5*(Y(I+1)+Y(I-1)+S-T)-Y(I)-U;                              194.
      B(I):=0.5*(Y(I+1)-Y(I-1)-S-T)-D(I)                                 195.
   END I;                                                                196.
   F(N1):=V;  E(N1):=E(N2):=D(N1):=D(N2):=0.0;                           197.
   C(N1):=C(N1+1)-10.0*V;  C(N2):=C(N2-1)+10.0*T;                        198.
   B(N1):=Y(N1+1)-Y(N1)-C(N1)-V;  B(N2):=Y(N2)-Y(N2-1)+C(N2)-T           199.
END QUINEQ;                                                              200.


PROCEDURE QUINDF(INTEGER VALUE N1,N2; REAL ARRAY X,Y,B,C,D,E,F(*));      201.
COMMENT QUINDF COMPUTES THE COEFFICIENTS OF A QUINTIC NATURAL SPLINE     202.
   S(X) FOR WHICH THE ORDINATES Y(I) AND THE FIRST DERIVATIVES B(I)      203.
   ARE SPECIFIED AT POINTS X(I), I = N1 THROUGH N2.  FOR XX IN           204.
   (X(I),X(I+1)) THE VALUE OF THE SPLINE FUNCTION S(XX) IS GIVEN         205.
```

```
    BY THE FIFTH DEGREE POLYNOMIAL:                                          206.
    S(XX) = ((((F(I)*T+E(I))*T+D(I))*T+C(I))*T+B(I))*T+Y(I)                  207.
    WITH T = XX - X(I).                                                      208.
    INPUT:                                                                   209.
      N1, N2  SUBSCRIPT OF FIRST AND LAST DATA POINT RESPECTIVELY,           210.
              IT IS REQUIRED THAT N2 > N1,                                   211.
      X,Y,B(N1::N2)  ARRAYS WITH X(I) AS ABSCISSA, Y(I) AS ORDINATE          212.
              AND B(I) AS FIRST DERIVATIVE AT THE I-TH DATA POINT.           213.
              THE ELEMENTS OF THE ARRAY X MUST BE STRICTLY MONOTONE          214.
              INCREASING OR DECREASING.                                      215.
    OUTPUT:                                                                  216.
      C,D,E,F(N1::N2)  ARRAYS COLLECTING THE COEFFICIENTS OF THE             217.
              QUINTIC NATURAL SPLINE S(XX) AS DESCRIBED ABOVE.               218.
              E(N2) AND F(N2) ARE NEITHER USED NOR ALTERED.  THE             219.
              ARRAYS C,D,E,F MUST ALWAYS BE DISTINCT;                        220.
  IF N2 > N1 THEN                                                            221.
    BEGIN                                                                    222.
    INTEGER  M2;                                                             223.
    REAL CC,G,H,HH,H2,HH2,P,PP,Q,QQ,R,RR;                                    224.
    M2:=N2-1;  CC:=HH:=PP:=QQ:=RR:=G:=0.0;                                   225.
    FOR I:=N1 STEP 1 UNTIL M2 DO                                             226.
    BEGIN                                                                    227.
       H:=1.0/(X(I+1)-X(I));  H2:=H*H;  D(I):=3.0*(HH+H) - G*HH;              228.
       P:=(Y(I+1)-Y(I))*H*H2;  Q:=(B(I+1)+B(I))*H2;                          229.
       R:=(B(I+1)-B(I))*H2;                                                  230.
       C(I):=CC:=10.0*(P-PP) - 5.0*(Q-QQ) + R + RR + G*CC;                   231.
       G:=H/D(I);  HH:=H;  HH2:=H2;  PP:=P;  QQ:=Q;  RR:=R                   232.
    END I;                                                                   233.
    C(N2):=(-10.0*PP + 5.0*QQ + RR + G*CC)/(3.0*HH - G*HH);                  234.
    FOR I:=M2 STEP -1 UNTIL N1 DO                                            235.
    BEGIN                                                                    236.
       D(I+1):=1.0/(X(I+1)-X(I));  C(I):=(C(I) + C(I+1)*D(I+1))/D(I)         237.
    END I;                                                                   238.
    FOR I:=N1 STEP 1 UNTIL M2 DO                                             239.
    BEGIN                                                                    240.
       H:=D(I+1);  H2:=H*H;                                                  241.
       P:=(Y(I+1)-Y(I))*H*H2 - B(I)*H2 -C(I)*H;                              242.
       Q:=(B(I+1)-B(I))*H2 - C(I)*(H+H);                                     243.
       R:=(C(I+1)-C(I))*H;                                                   244.
       G:=Q - 3.0*P;  RR:=R - 3.0*(P+G);  QQ:= -RR - RR + G;                 245.
       F(I):=RR*H2;  E(I):=QQ*H;  D(I):= -RR - QQ + P                        246.
    END I;                                                                   247.
    D(N2):=0.0                                                               248.
  END QUINDF;                                                                249.
```


REMARK ON ALGORITHM 507

Procedures for Quintic Natural Spline Interpolation
[J.G. Herriot and C.H. Reinsch, *ACM Trans. Math. Softw. 2*, 3 (June 1976), 281–289.]

R.J. Hanson [Received 10 March 1982; accepted 10 March 1982]
Numerical Mathematics Division 5642, Sandia National Laboratories, Albuquerque, NM 87185

Line number 82. of Algorithm 507

IF R ¬= 0.0 THEN D(M) := D(M) + 6.0*R*R2/(QR*QR)

is changed to the mathematically equivalent line

IF ABS(R) > 0.0 THEN D(M) := D(M) + 6.0*R*R2/(QR*QR)

The change is made for this reason: the character "¬" will often not transmit correctly in the *Collected Algorithms from ACM* distribution process.

ALGORITHM 508
Matrix Bandwidth and Profile Reduction [F1]

H. L. CRANE JR.
Comptek Research, Inc.
NORMAN E. GIBBS, WILLIAM G. POOLE JR., and PAUL K. STOCKMEYER
College of William and Mary

Key Words and Phrases: bandwidth reduction, diameter of a graph, profile reduction, sparse matrices
CR Categories: 5.14, 5.32
Language: Fortran

DESCRIPTION

Introduction

This program, REDUCE, reduces the bandwidth and profile of sparse symmetric matrices, using row and corresponding column permutations. It is a realization of the algorithm described by the authors in [4]. It was extensively tested and compared with several other programs [5] and was found to be considerably faster than the others, generally superior for bandwidth reduction and as satisfactory as any other for profile reduction.

Outline of the Method

Only an outline of the algorithm is given here; a detailed description can be found in [4]. The algorithm can best be described in terms of the *adjacency graph* G, which has the characteristic that there is an edge in G between vertices v_i and v_j if and only if $a_{ij} \neq 0$ and $i \neq j$.

Step 1. Find the endpoints of a *pseudodiameter*, i.e. a pair of vertices that are at nearly maximal distance apart. This is done by a finite, iterative process of determining a vertex that is a maximum distance away from a given vertex.

Step 2. Given pseudodiameter endpoints u and v of distance k apart, partition the set of vertices into *levels* $L_1, L_2, \ldots, L_k$ such that adjacent vertices in G are in the same or adjacent levels and such that $\max_i | L_i |$ is nearly minimized.

Step 3. Number the vertices of G, level by level, beginning at an endpoint of the pseudodiameter.

Matrix Data Structure

Sparse matrices are typically stored in some compact form which takes advantage of the sparsity. The data structure assumed here is one which is commonly used in

Received 17 May 1975 and 3 March 1976.

This research was supported in part by the Office of Naval Research under Contract N00014-73-A-0374-0001, NR044-459, and in part by the National Aeronautics and Space Administration under Grant NGR 47-102-001.
Authors' addresses: H.L. Crane Jr., Comptek Research, Inc., Hyattsville, MD 20782; N.E. Gibbs, W.G. Poole Jr., P.K. Stockmeyer, Department of Mathematics, College of William and Mary, Williamsburg, VA 23185.

bandwidth and profile schemes, e.g. [1], [2], [3], and [6]. Subroutine REDUCE accepts as input a *connection table*, C, representing the indices of the nonzero elements of the $n \times n$ matrix A. The connection table has n rows and m columns where m is the number of off-diagonal nonzero elements in the row which has a maximum number of off-diagonal nonzero elements (i.e. the maximum degree of the graph G). The entries in row i of C are the column indices of the nonzero elements in row i of the matrix A. For example, if

$$A = \begin{bmatrix} X & X & 0 & 0 & X \\ X & X & 0 & X & X \\ 0 & 0 & X & X & 0 \\ 0 & X & X & X & 0 \\ X & X & 0 & 0 & X \end{bmatrix}$$

where X represents a nonzero element, then

$$C = \begin{bmatrix} 2 & 5 & 0 \\ 1 & 4 & 5 \\ 4 & 0 & 0 \\ 2 & 3 & 0 \\ 1 & 2 & 0 \end{bmatrix}.$$

The order of indices in a row of C is immaterial. The nonzero elements of the matrix A are never needed, only their indices.

Test Results

REDUCE was tested on an IBM System/360 (model 50) computer using the Fortran IV G and H compilers and on a CDC 6600 computer using the Fortran (RUN) and Fortran extended (FTN) compilers.

In another paper [5], the authors compared the execution times, bandwidths, and profiles produced by REDUCE with those of five other programs on a wide range of problems. REDUCE typically produced the smallest bandwidths; it produced profiles which were on the average as small as those for any other program; and REDUCE was faster than all the others by at least an order of magnitude.

REFERENCES

1. Collins, R.J. Bandwidth reduction by automatic renumbering. *Int. J. Numer. Meth. Engrg. 6* (1973), 345–356.
2. Cuthill, E.H., and McKee, J. Reducing the bandwidth of sparse symmetric matrices. Proc. ACM 24th Nat. Conf., ACM, New York, 1969, pp. 157–172.
3. Everstine, G.C. The BANDIT computer program for the reduction of matrix bandwidth for NASTRAN. Rep. 3827, Naval Ship Res. and Develop. Center, Bethesda, Md., 1972.
4. Gibbs, N.E., Poole, W.G. Jr., and Stockmeyer, P.K. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM J. Numer. Anal. 13*, 2 (April 1976), 235–251.
5. Gibbs, N.E., Poole, W.G. Jr., and Stockmeyer, P.K. A comparison of several bandwidth and profile reduction algorithms. *ACM Trans. Math. Software 2*, 4 (Dec. 1976), 322–330.
6. Wang, P.T.R. Bandwidth minimization, reducibility, decomposition, and triangularization of sparse matrices. Ph.D. Diss., Dep. Comptr. and Information Sci., Ohio State U., Columbus, Ohio, 1973.

ALGORITHM

```
      SUBROUTINE REDUCE(NDSTK, NR, IOLD, RENUM, NDEG, LVL, LVLS1,      RED   1Ø
     * LVLS2, CCSTOR, IBW2, IPF2)                                      RED   2Ø
C  SUBROUTINE REDUCE DETERMINES A ROW AND COLUMN PERMUTATION WHICH,     RED   3Ø
C  WHEN APPLIED TO A GIVEN SPARSE MATRIX, PRODUCES A PERMUTED           RED   4Ø
C  MATRIX WITH A SMALLER BANDWIDTH AND PROFILE.                         RED   5Ø
C  THE INPUT ARRAY IS A CONNECTION TABLE WHICH REPRESENTS THE           RED   6Ø
C  INDICES OF THE NONZERO ELEMENTS OF THE MATRIX, A.  THE ALGO-         RED   7Ø
C  RITHM IS DESCRIBED IN TERMS OF THE ADJACENCY GRAPH WHICH             RED   8Ø
C  HAS THE CHARACTERISTIC THAT THERE IS AN EDGE (CONNECTION)            RED   9Ø
C  BETWEEN NODES I AND J IF A(I,J) .NE. Ø AND I .NE. J.                 RED  1ØØ
C  DIMENSIONING INFORMATION--THE FOLLOWING INTEGER ARRAYS MUST BE       RED  11Ø
C  DIMENSIONED IN THE CALLING ROUTINE.                                  RED  12Ø
C    NDSTK(NR,D1)          D1 IS .GE. MAXIMUM DEGREE OF ALL NODES.     RED  13Ø
C    IOLD(D2)              D2 AND NR ARE .GE. THE TOTAL NUMBER OF      RED  14Ø
C    RENUM(D2+1)           NODES IN THE GRAPH.                         RED  15Ø
```

```
C     NDEG(D2)            STORAGE REQUIREMENTS CAN BE SIGNIFICANTLY      RED  16Ø
C     LVL(D2)             DECREASED FOR IBM 36Ø AND 37Ø COMPUTERS        RED  17Ø
C     LVLS1(D2)           BY REPLACING INTEGER NDSTK BY                  RED  18Ø
C     LVLS2(D2)           INTEGER*2 NDSTK IN SUBROUTINES REDUCE,         RED  19Ø
C     CCSTOR(D2)          DGREE, FNDIAM, TREE AND NUMBER.                RED  2ØØ
C   COMMON INFORMATION--THE FOLLOWING COMMON BLOCK MUST BE IN THE        RED  21Ø
C   CALLING ROUTINE.                                                     RED  22Ø
C     COMMON/GRA/N,IDPTH,IDEG                                            RED  23Ø
C   EXPLANATION OF INPUT VARIABLES--                                     RED  24Ø
C     NDSTK-      CONNECTION TABLE REPRESENTING GRAPH.                   RED  25Ø
C                 NDSTK(I,J)=NODE NUMBER OF JTH CONNECTION TO NODE       RED  26Ø
C                 NUMBER I.  A CONNECTION OF A NODE TO ITSELF IS NOT     RED  27Ø
C                 LISTED.  EXTRA POSITIONS MUST HAVE ZERO FILL.          RED  28Ø
C     NR-         ROW DIMENSION ASSIGNED NDSTK IN CALLING PROGRAM.       RED  29Ø
C     IOLD(I)-    NUMBERING OF ITH NODE UPON INPUT.                      RED  3ØØ
C                 IF NO NUMBERING EXISTS THEN IOLD(I)=I.                 RED  31Ø
C     N-          NUMBER OF NODES IN GRAPH (EQUAL TO ORDER OF MATRIX).   RED  32Ø
C     IDEG-       MAXIMUM DEGREE OF ANY NODE IN THE GRAPH.               RED  33Ø
C   EXPLANATION OF OUTPUT VARIABLES--                                    RED  34Ø
C     RENUM(I)-   THE NEW NUMBER FOR THE ITH NODE.                       RED  35Ø
C     NDEG(I)-    THE DEGREE OF THE ITH NODE.                            RED  36Ø
C     IBW2-       THE BANDWIDTH AFTER RENUMBERING.                       RED  37Ø
C     IPF2-       THE PROFILE AFTER RENUMBERING.                         RED  38Ø
C     IDPTH-      NUMBER OF LEVELS IN REDUCE LEVEL STRUCTURE.            RED  39Ø
C   THE FOLLOWING ONLY HAVE MEANING IF THE GRAPH WAS CONNECTED--         RED  4ØØ
C     LVL(I)-     INDEX INTO LVLS1 TO THE FIRST NODE IN LEVEL I.         RED  41Ø
C                 LVL(I+1)-LVL(I)= NUMBER OF NODES IN ITH LEVEL          RED  42Ø
C     LVLS1-      NODE NUMBERS LISTED BY LEVEL.                          RED  43Ø
C     LVLS2(I)-   THE LEVEL ASSIGNED TO NODE I BY REDUCE.                RED  44Ø
C   WORKING STORAGE VARIABLE--                                           RED  45Ø
C     CCSTOR                                                             RED  46Ø
C   LOCAL STORAGE--                                                      RED  47Ø
C     COMMON/CC/-SUBROUTINES REDUCE, SORT2 AND PIKLVL ASSUME THAT        RED  48Ø
C                 THE GRAPH HAS AT MOST 5Ø CONNECTED COMPONENTS.         RED  49Ø
C                 SUBROUTINE FNDIAM ASSUMES THAT THERE ARE AT MOST       RED  5ØØ
C                 1ØØ NODES IN THE LAST LEVEL.                           RED  51Ø
C     COMMON/LVLW/-SUBROUTINES SETUP AND PIKLVL ASSUME THAT THERE        RED  52Ø
C                 ARE AT MOST 1ØØ LEVELS.                                RED  53Ø
C USE INTEGER*2 NDSTK  WITH AN IBM 36Ø OR 37Ø.                           RED  54Ø
      INTEGER NDSTK                                                      RED  55Ø
      INTEGER STNODE, RVNODE, RENUM, XC, SORT2, STNUM, CCSTOR,           RED  56Ø
     * SIZE, STPT, SBNUM                                                RED  57Ø
      COMMON /GRA/ N, IDPTH, IDEG                                        RED  58Ø
C IT IS ASSUMED THAT THE GRAPH HAS AT MOST 5Ø CONNECTED COMPONENTS.      RED  59Ø
      COMMON /CC/ XC, SIZE(5Ø), STPT(5Ø)                                 RED  6ØØ
      COMMON /LVLW/ NHIGH(1ØØ), NLOW(1ØØ), NACUM(1ØØ)                    RED  61Ø
      DIMENSION CCSTOR(1), IOLD(1)                                       RED  62Ø
      DIMENSION NDSTK(NR,1), LVL(1), LVLS1(1), LVLS2(1), RENUM(1),       RED  63Ø
     * NDEG(1)                                                          RED  64Ø
      IBW2 = Ø                                                           RED  65Ø
      IPF2 = Ø                                                           RED  66Ø
C SET RENUM(I)=Ø FOR ALL I TO INDICATE NODE I IS UNNUMBERED              RED  67Ø
      DO 1Ø I=1,N                                                        RED  68Ø
        RENUM(I) = Ø                                                     RED  69Ø
   1Ø CONTINUE                                                           RED  7ØØ
C COMPUTE DEGREE OF EACH NODE AND ORIGINAL BANDWIDTH AND PROFILE         RED  71Ø
      CALL DGREE(NDSTK, NR, NDEG, IOLD, IBW1, IPF1)                      RED  72Ø
C SBNUM= LOW END OF AVAILABLE NUMBERS FOR RENUMBERING                    RED  73Ø
C STNUM= HIGH END OF AVAILABLE NUMBERS FOR RENUMBERING                   RED  74Ø
      SBNUM = 1                                                          RED  75Ø
      STNUM = N                                                          RED  76Ø
C NUMBER THE NODES OF DEGREE ZERO                                        RED  77Ø
      DO 2Ø I=1,N                                                        RED  78Ø
        IF (NDEG(I).GT.Ø) GO TO 2Ø                                       RED  79Ø
        RENUM(I) = STNUM                                                 RED  8ØØ
        STNUM = STNUM - 1                                                RED  81Ø
   2Ø CONTINUE                                                           RED  82Ø
C FIND AN UNNUMBERED NODE OF MIN DEGREE TO START ON                      RED  83Ø
   3Ø LOWDG = IDEG + 1                                                   RED  84Ø
      NFLG = 1                                                           RED  85Ø
      ISDIR = 1                                                          RED  86Ø
      DO 4Ø I=1,N                                                        RED  87Ø
        IF (NDEG(I).GE.LOWDG) GO TO 4Ø                                   RED  88Ø
        IF (RENUM(I).GT.Ø) GO TO 4Ø                                      RED  89Ø
        LOWDG = NDEG(I)                                                  RED  9ØØ
        STNODE = I                                                       RED  91Ø
```

```
   4Ø CONTINUE                                                          RED  92Ø
C FIND PSEUDO-DIAMETER AND ASSOCIATED LEVEL STRUCTURES.                 RED  93Ø
C STNODE AND RVNODE ARE THE ENDS OF THE DIAM AND LVLS1 AND LVLS2        RED  94Ø
C ARE THE RESPECTIVE LEVEL STRUCTURES.                                  RED  95Ø
      CALL FNDIAM(STNODE, RVNODE, NDSTK, NR, NDEG, LVL, LVLS1,          RED  96Ø
     * LVLS2, CCSTOR, IDFLT)                                           RED  97Ø
      IF (NDEG(STNODE).LE.NDEG(RVNODE)) GO TO 5Ø                        RED  98Ø
C NFLG INDICATES THE END TO BEGIN NUMBERING ON                          RED  99Ø
      NFLG = -1                                                         RED 1ØØØ
      STNODE = RVNODE                                                   RED 1Ø1Ø
   5Ø CALL SETUP(LVL, LVLS1, LVLS2)                                     RED 1Ø2Ø
C FIND ALL THE CONNECTED COMPONENTS  (XC COUNTS THEM)                   RED 1Ø3Ø
      XC = Ø                                                            RED 1Ø4Ø
      LROOT = 1                                                         RED 1Ø5Ø
      LVLN = 1                                                          RED 1Ø6Ø
      DO 6Ø I=1,N                                                       RED 1Ø7Ø
        IF (LVL(I).NE.Ø) GO TO 6Ø                                       RED 1Ø8Ø
        XC = XC + 1                                                     RED 1Ø9Ø
        STPT(XC) = LROOT                                                RED 11ØØ
        CALL TREE(I, NDSTK, NR, LVL, CCSTOR, NDEG, LVLWTH, LVLBOT,      RED 111Ø
     *   LVLN, MAXLW, N)                                                RED 112Ø
        SIZE(XC) = LVLBOT + LVLWTH - LROOT                              RED 113Ø
        LROOT = LVLBOT + LVLWTH                                         RED 114Ø
        LVLN = LROOT                                                    RED 115Ø
   6Ø CONTINUE                                                          RED 116Ø
      IF (SORT2(DMY).EQ.Ø) GO TO 7Ø                                     RED 117Ø
      CALL PIKLVL(LVLS1, LVLS2, CCSTOR, IDFLT, ISDIR)                   RED 118Ø
C ON RETURN FROM PIKLVL, ISDIR INDICATES THE DIRECTION THE LARGEST      RED 119Ø
C COMPONENT FELL.  ISDIR IS MODIFIED NOW TO INDICATE THE NUMBERING      RED 12ØØ
C DIRECTION.  NUM IS SET TO THE PROPER VALUE FOR THIS DIRECTION.        RED 121Ø
   7Ø ISDIR = ISDIR*NFLG                                                RED 122Ø
      NUM = SBNUM                                                       RED 123Ø
      IF (ISDIR.LT.Ø) NUM = STNUM                                       RED 124Ø
      CALL NUMBER(STNODE, NUM, NDSTK, LVLS2, NDEG, RENUM, LVLS1,        RED 125Ø
     * LVL, NR, NFLG, IBW2, IPF2, CCSTOR, ISDIR)                       RED 126Ø
C UPDATE STNUM OR SBNUM AFTER NUMBERING                                 RED 127Ø
      IF (ISDIR.LT.Ø) STNUM = NUM                                       RED 128Ø
      IF (ISDIR.GT.Ø) SBNUM = NUM                                       RED 129Ø
      IF (SBNUM.LE.STNUM) GO TO 3Ø                                      RED 13ØØ
      IF (IBW2.LE.IBW1) RETURN                                          RED 131Ø
C IF ORIGINAL NUMBERING IS BETTER THAN NEW ONE, SET UP TO RETURN IT     RED 132Ø
      DO 8Ø I=1,N                                                       RED 133Ø
        RENUM(I) = IOLD(I)                                              RED 134Ø
   8Ø CONTINUE                                                          RED 135Ø
      IBW2 = IBW1                                                       RED 136Ø
      IPF2 = IPF1                                                       RED 137Ø
      RETURN                                                            RED 138Ø
      END                                                               RED 139Ø


      SUBROUTINE DGREE(NDSTK, NR, NDEG, IOLD, IBW1, IPF1)               DGR   1Ø
C  DGREE COMPUTES THE DEGREE OF EACH NODE IN NDSTK AND STORES           DGR   2Ø
C  IT IN THE ARRAY NDEG.  THE BANDWIDTH AND PROFILE FOR THE ORIGINAL    DGR   3Ø
C  OR INPUT RENUMBERING OF THE GRAPH IS COMPUTED ALSO.                  DGR   4Ø
C USE INTEGER*2 NDSTK  WITH AN IBM 36Ø OR 37Ø.                          DGR   5Ø
      INTEGER NDSTK                                                     DGR   6Ø
      COMMON /GRA/ N, IDPTH, IDEG                                       DGR   7Ø
      DIMENSION NDSTK(NR,1), NDEG(1), IOLD(1)                           DGR   8Ø
      IBW1 = Ø                                                          DGR   9Ø
      IPF1 = Ø                                                          DGR  1ØØ
      DO 4Ø I=1,N                                                       DGR  11Ø
        NDEG(I) = Ø                                                     DGR  12Ø
        IRW = Ø                                                         DGR  13Ø
        DO 2Ø J=1,IDEG                                                  DGR  14Ø
          ITST = NDSTK(I,J)                                             DGR  15Ø
          IF (ITST) 3Ø, 3Ø, 1Ø                                          DGR  16Ø
   1Ø     NDEG(I) = NDEG(I) + 1                                         DGR  17Ø
          IDIF = IOLD(I) - IOLD(ITST)                                   DGR  18Ø
          IF (IRW.LT.IDIF) IRW = IDIF                                   DGR  19Ø
   2Ø   CONTINUE                                                        DGR  2ØØ
   3Ø   IPF1 = IPF1 + IRW                                               DGR  21Ø
        IF (IRW.GT.IBW1) IBW1 = IRW                                     DGR  22Ø
   4Ø CONTINUE                                                          DGR  23Ø
      RETURN                                                            DGR  24Ø
      END                                                               DGR  25Ø
```

```
      SUBROUTINE FNDIAM(SND1, SND2, NDSTK, NR, NDEG, LVL, LVLS1,        FND   1Ø
     * LVLS2, IWK, IDFLT)                                               FND   2Ø
C  FNDIAM IS THE CONTROL PROCEDURE FOR FINDING THE PSEUDO-DIAMETER OF   FND   3Ø
C  NDSTK AS WELL AS THE LEVEL STRUCTURE FROM EACH END                   FND   4Ø
C  SND1-        ON INPUT THIS IS THE NODE NUMBER OF THE FIRST           FND   5Ø
C               ATTEMPT AT FINDING A DIAMETER.  ON OUTPUT IT            FND   6Ø
C               CONTAINS THE ACTUAL NUMBER USED.                        FND   7Ø
C  SND2-        ON OUTPUT CONTAINS OTHER END OF DIAMETER                FND   8Ø
C  LVLS1-       ARRAY CONTAINING LEVEL STRUCTURE WITH SND1 AS ROOT      FND   9Ø
C  LVLS2-       ARRAY CONTAINING LEVEL STRUCTURE WITH SND2 AS ROOT      FND  1ØØ
C  IDFLT-       FLAG USED IN PICKING FINAL LEVEL STRUCTURE, SET         FND  11Ø
C               =1 IF WIDTH OF LVLS1 .LE. WIDTH OF LVLS2, OTHERWISE =2  FND  12Ø
C  LVL,IWK-     WORKING STORAGE                                         FND  13Ø
C USE INTEGER*2 NDSTK  WITH AN IBM 36Ø OR 37Ø.                         FND  14Ø
      INTEGER NDSTK                                                     FND  15Ø
      INTEGER FLAG, SND, SND1, SND2                                     FND  16Ø
      COMMON /GRA/ N, IDPTH, IDEG                                       FND  17Ø
C IT IS ASSUMED THAT THE LAST LEVEL HAS AT MOST 1ØØ NODES.             FND  18Ø
      COMMON /CC/ NDLST(1ØØ)                                            FND  19Ø
      DIMENSION NDSTK(NR,1), NDEG(1), LVL(1), LVLS1(1), LVLS2(1),       FND  2ØØ
     * IWK(1)                                                           FND  21Ø
      FLAG = Ø                                                          FND  22Ø
      MTW2 = N                                                          FND  23Ø
      SND = SND1                                                        FND  24Ø
C ZERO LVL TO INDICATE ALL NODES ARE AVAILABLE TO TREE                 FND  25Ø
   1Ø DO 2Ø I=1,N                                                       FND  26Ø
        LVL(I) = Ø                                                      FND  27Ø
   2Ø CONTINUE                                                          FND  28Ø
      LVLN = 1                                                          FND  29Ø
C DROP A TREE FROM SND                                                 FND  3ØØ
      CALL TREE(SND, NDSTK, NR, LVL, IWK, NDEG, LVLWTH, LVLBOT,         FND  31Ø
     * LVLN, MAXLW, MTW2)                                              FND  32Ø
      IF (FLAG.GE.1) GO TO 5Ø                                           FND  33Ø
      FLAG = 1                                                          FND  34Ø
   3Ø IDPTH = LVLN - 1                                                  FND  35Ø
      MTW1 = MAXLW                                                      FND  36Ø
C COPY LEVEL STRUCTURE INTO LVLS1                                      FND  37Ø
      DO 4Ø I=1,N                                                       FND  38Ø
        LVLS1(I) = LVL(I)                                               FND  39Ø
   4Ø CONTINUE                                                          FND  4ØØ
      NDXN = 1                                                          FND  41Ø
      NDXL = Ø                                                          FND  42Ø
      MTW2 = N                                                          FND  43Ø
C SORT LAST LEVEL BY DEGREE  AND STORE IN NDLST                        FND  44Ø
      CALL SORTDG(NDLST, IWK(LVLBOT), NDXL, LVLWTH, NDEG)               FND  45Ø
      SND = NDLST(1)                                                    FND  46Ø
      GO TO 1Ø                                                          FND  47Ø
   5Ø IF (IDPTH.GE.LVLN-1) GO TO 6Ø                                     FND  48Ø
C START AGAIN WITH NEW STARTING NODE                                   FND  49Ø
      SND1 = SND                                                        FND  5ØØ
      GO TO 3Ø                                                          FND  51Ø
   6Ø IF (MAXLW.GE.MTW2) GO TO 8Ø                                       FND  52Ø
      MTW2 = MAXLW                                                      FND  53Ø
      SND2 = SND                                                        FND  54Ø
C STORE NARROWEST REVERSE LEVEL STRUCTURE IN LVLS2                     FND  55Ø
      DO 7Ø I=1,N                                                       FND  56Ø
        LVLS2(I) = LVL(I)                                               FND  57Ø
   7Ø CONTINUE                                                          FND  58Ø
   8Ø IF (NDXN.EQ.NDXL) GO TO 9Ø                                        FND  59Ø
C TRY NEXT NODE IN NDLST                                               FND  6ØØ
      NDXN = NDXN + 1                                                   FND  61Ø
      SND = NDLST(NDXN)                                                 FND  62Ø
      GO TO 1Ø                                                          FND  63Ø
   9Ø IDFLT = 1                                                         FND  64Ø
      IF (MTW2.LE.MTW1) IDFLT = 2                                       FND  65Ø
      RETURN                                                            FND  66Ø
      END                                                               FND  67Ø


      SUBROUTINE TREE(IROOT, NDSTK, NR, LVL, IWK, NDEG, LVLWTH,         TRE   1Ø
     * LVLBOT, LVLN, MAXLW, IBORT)                                      TRE   2Ø
C  TREE DROPS A TREE IN NDSTK FROM IROOT                                TRE   3Ø
C  LVL-         ARRAY INDICATING AVAILABLE NODES IN NDSTK WITH ZERO     TRE   4Ø
C               ENTRIES. TREE ENTERS LEVEL NUMBERS ASSIGNED             TRE   5Ø
```

```
C                DURING EXECUTION OF THIS PROCEDURE                        TRE   6Ø
C  IWK-          ON OUTPUT CONTAINS NODE NUMBERS USED IN TREE              TRE   7Ø
C                ARRANGED BY LEVELS (IWK(LVLN) CONTAINS IROOT              TRE   8Ø
C                AND IWK(LVLBOT+LVLWTH-1) CONTAINS LAST NODE ENTERED)      TRE   9Ø
C  LVLWTH-       ON OUTPUT CONTAINS WIDTH OF LAST LEVEL                    TRE  1ØØ
C  LVLBOT-       ON OUTPUT CONTAINS INDEX INTO IWK OF FIRST                TRE  11Ø
C                NODE IN LAST LEVEL                                        TRE  12Ø
C  MAXLW-        ON OUTPUT CONTAINS THE MAXIMUM LEVEL WIDTH                TRE  13Ø
C  LVLN-         ON INPUT THE FIRST AVAILABLE LOCATION IN IWK              TRE  14Ø
C                USUALLY ONE BUT IF IWK IS USED TO STORE PREVIOUS          TRE  15Ø
C                CONNECTED COMPONENTS, LVLN IS NEXT AVAILABLE LOCATION.    TRE  16Ø
C                ON OUTPUT THE TOTAL NUMBER OF LEVELS + 1                  TRE  17Ø
C  IBORT-        INPUT PARAM WHICH TRIGGERS EARLY RETURN IF                TRE  18Ø
C                MAXLW BECOMES .GE. IBORT                                  TRE  19Ø
C USE INTEGER*2 NDSTK  WITH AN IBM 36Ø OR 37Ø.                             TRE  2ØØ
      INTEGER NDSTK                                                        TRE  21Ø
      DIMENSION NDSTK(NR,1), LVL(1), IWK(1), NDEG(1)                       TRE  22Ø
      MAXLW = Ø                                                            TRE  23Ø
      ITOP = LVLN                                                          TRE  24Ø
      INOW = LVLN                                                          TRE  25Ø
      LVLBOT = LVLN                                                        TRE  26Ø
      LVLTOP = LVLN + 1                                                    TRE  27Ø
      LVLN = 1                                                             TRE  28Ø
      LVL(IROOT) = 1                                                       TRE  29Ø
      IWK(ITOP) = IROOT                                                    TRE  3ØØ
   1Ø LVLN = LVLN + 1                                                      TRE  31Ø
   2Ø IWKNOW = IWK(INOW)                                                   TRE  32Ø
      NDROW = NDEG(IWKNOW)                                                 TRE  33Ø
      DO 3Ø J=1,NDROW                                                      TRE  34Ø
        ITEST = NDSTK(IWKNOW,J)                                            TRE  35Ø
        IF (LVL(ITEST).NE.Ø) GO TO 3Ø                                      TRE  36Ø
        LVL(ITEST) = LVLN                                                  TRE  37Ø
        ITOP = ITOP + 1                                                    TRE  38Ø
        IWK(ITOP) = ITEST                                                  TRE  39Ø
   3Ø CONTINUE                                                             TRE  4ØØ
      INOW = INOW + 1                                                      TRE  41Ø
      IF (INOW.LT.LVLTOP) GO TO 2Ø                                         TRE  42Ø
      LVLWTH = LVLTOP - LVLBOT                                             TRE  43Ø
      IF (MAXLW.LT.LVLWTH) MAXLW = LVLWTH                                  TRE  44Ø
      IF (MAXLW.GE.IBORT) RETURN                                           TRE  45Ø
      IF (ITOP.LT.LVLTOP) RETURN                                           TRE  46Ø
      LVLBOT = INOW                                                        TRE  47Ø
      LVLTOP = ITOP + 1                                                    TRE  48Ø
      GO TO 1Ø                                                             TRE  49Ø
      END                                                                  TRE  5ØØ


      SUBROUTINE SORTDG(STK1, STK2, X1, X2, NDEG)                          SOR   1Ø
C SORTDG SORTS STK2 BY DEGREE OF THE NODE AND ADDS IT TO THE END           SOR   2Ø
C OF STK1 IN ORDER OF LOWEST TO HIGHEST DEGREE.  X1 AND X2 ARE THE         SOR   3Ø
C NUMBER OF NODES IN STK1 AND STK2 RESPECTIVELY.                           SOR   4Ø
      INTEGER X1, X2, STK1, STK2, TEMP                                     SOR   5Ø
      COMMON /GRA/ N, IDPTH, IDEG                                          SOR   6Ø
      DIMENSION NDEG(1), STK1(1), STK2(1)                                  SOR   7Ø
      IND = X2                                                             SOR   8Ø
   1Ø ITEST = Ø                                                            SOR   9Ø
      IND = IND - 1                                                        SOR  1ØØ
      IF (IND.LT.1) GO TO 3Ø                                               SOR  11Ø
      DO 2Ø I=1,IND                                                        SOR  12Ø
        J = I + 1                                                          SOR  13Ø
        ISTK2 = STK2(I)                                                    SOR  14Ø
        JSTK2 = STK2(J)                                                    SOR  15Ø
        IF (NDEG(ISTK2).LE.NDEG(JSTK2)) GO TO 2Ø                           SOR  16Ø
        ITEST = 1                                                          SOR  17Ø
        TEMP = STK2(I)                                                     SOR  18Ø
        STK2(I) = STK2(J)                                                  SOR  19Ø
        STK2(J) = TEMP                                                     SOR  2ØØ
   2Ø CONTINUE                                                             SOR  21Ø
      IF (ITEST.EQ.1) GO TO 1Ø                                             SOR  22Ø
   3Ø DO 4Ø I=1,X2                                                         SOR  23Ø
        X1 = X1 + 1                                                        SOR  24Ø
        STK1(X1) = STK2(I)                                                 SOR  25Ø
   4Ø CONTINUE                                                             SOR  26Ø
      RETURN                                                               SOR  27Ø
      END                                                                  SOR  28Ø
```

```
      SUBROUTINE SETUP(LVL, LVLS1, LVLS2)                                SET  1Ø
C SETUP COMPUTES THE REVERSE LEVELING INFO FROM LVLS2 AND STORES          SET  2Ø
C IT INTO LVLS2.  NACUM(I) IS INITIALIZED TO NODES/ITH LEVEL FOR NODES    SET  3Ø
C ON THE PSEUDO-DIAMETER OF THE GRAPH.  LVL IS INITIALIZED TO NON-        SET  4Ø
C ZERO FOR NODES ON THE PSEUDO-DIAM AND NODES IN A DIFFERENT              SET  5Ø
C COMPONENT OF THE GRAPH.                                                 SET  6Ø
      COMMON /GRA/ N, IDPTH, IDEG                                         SET  7Ø
C IT IS ASSUMED THAT THERE ARE AT MOST 1ØØ LEVELS.                        SET  8Ø
      COMMON /LVLW/ NHIGH(1ØØ), NLOW(1ØØ), NACUM(1ØØ)                     SET  9Ø
      DIMENSION LVL(1), LVLS1(1), LVLS2(1)                                SET 1ØØ
      DO 1Ø I=1,IDPTH                                                     SET 11Ø
        NACUM(I) = Ø                                                      SET 12Ø
   1Ø CONTINUE                                                            SET 13Ø
      DO 3Ø I=1,N                                                         SET 14Ø
        LVL(I) = 1                                                        SET 15Ø
        LVLS2(I) = IDPTH + 1 - LVLS2(I)                                   SET 16Ø
        ITEMP = LVLS2(I)                                                  SET 17Ø
        IF (ITEMP.GT.IDPTH) GO TO 3Ø                                      SET 18Ø
        IF (ITEMP.NE.LVLS1(I)) GO TO 2Ø                                   SET 19Ø
        NACUM(ITEMP) = NACUM(ITEMP) + 1                                   SET 2ØØ
        GO TO 3Ø                                                          SET 21Ø
   2Ø   LVL(I) = Ø                                                        SET 22Ø
   3Ø CONTINUE                                                            SET 23Ø
      RETURN                                                              SET 24Ø
      END                                                                 SET 25Ø


      INTEGER FUNCTION SORT2(DMY)                                         SOR  1Ø
C SORT2 SORTS SIZE AND STPT INTO DESCENDING ORDER ACCORDING TO             SOR  2Ø
C VALUES OF SIZE. XC=NUMBER OF ENTRIES IN EACH ARRAY                      SOR  3Ø
      INTEGER TEMP, CCSTOR, SIZE, STPT, XC                                SOR  4Ø
C IT IS ASSUMED THAT THE GRAPH HAS AT MOST 5Ø CONNECTED COMPONENTS.       SOR  5Ø
      COMMON /CC/ XC, SIZE(5Ø), STPT(5Ø)                                  SOR  6Ø
      SORT2 = Ø                                                           SOR  7Ø
      IF (XC.EQ.Ø) RETURN                                                 SOR  8Ø
      SORT2 = 1                                                           SOR  9Ø
      IND = XC                                                            SOR 1ØØ
   1Ø ITEST = Ø                                                           SOR 11Ø
      IND = IND - 1                                                       SOR 12Ø
      IF (IND.LT.1) RETURN                                                SOR 13Ø
      DO 2Ø I=1,IND                                                       SOR 14Ø
        J = I + 1                                                         SOR 15Ø
        IF (SIZE(I).GE.SIZE(J)) GO TO 2Ø                                  SOR 16Ø
        ITEST = 1                                                         SOR 17Ø
        TEMP = SIZE(I)                                                    SOR 18Ø
        SIZE(I) = SIZE(J)                                                 SOR 19Ø
        SIZE(J) = TEMP                                                    SOR 2ØØ
        TEMP = STPT(I)                                                    SOR 21Ø
        STPT(I) = STPT(J)                                                 SOR 22Ø
        STPT(J) = TEMP                                                    SOR 23Ø
   2Ø CONTINUE                                                            SOR 24Ø
      IF (ITEST.EQ.1) GO TO 1Ø                                            SOR 25Ø
      RETURN                                                              SOR 26Ø
      END                                                                 SOR 27Ø


      SUBROUTINE PIKLVL(LVLS1, LVLS2, CCSTOR, IDFLT, ISDIR)               PIK  1Ø
C PIKLVL CHOOSES THE LEVEL STRUCTURE  USED IN NUMBERING GRAPH             PIK  2Ø
C LVLS1-       ON INPUT CONTAINS FORWARD LEVELING INFO                    PIK  3Ø
C LVLS2-       ON INPUT CONTAINS REVERSE LEVELING INFO                    PIK  4Ø
C              ON OUTPUT THE FINAL LEVEL STRUCTURE CHOSEN                 PIK  5Ø
C CCSTOR-      ON INPUT CONTAINS CONNECTED COMPONENT INFO                 PIK  6Ø
C IDFLT-       ON INPUT =1 IF WDTH LVLS1.LE.WDTH LVLS2, =2 OTHERWISE      PIK  7Ø
C NHIGH        KEEPS TRACK OF LEVEL WIDTHS FOR HIGH NUMBERING             PIK  8Ø
C NLOW-        KEEPS TRACK OF LEVEL WIDTHS FOR LOW NUMBERING              PIK  9Ø
C NACUM-       KEEPS TRACK OF LEVEL WIDTHS FOR CHOSEN LEVEL STRUCTURE     PIK 1ØØ
C XC-          NUMBER OF CONNECTED COMPONENTS                             PIK 11Ø
C SIZE(I)-     SIZE ρF ITH CONNECTED COMPONENT                            PIK 12Ø
C STPT(I)-     INDEX INTO CCSTORE OF 1ST NODE IN ITH CON COMPT            PIK 13Ø
C ISDIR-       FLAG WHICH INDICATES WHICH WAY THE LARGEST CONNECTED       PIK 14Ø
C              COMPONENT FELL.  =+1 IF LOW AND -1 IF HIGH                 PIK 15Ø
      INTEGER CCSTOR, SIZE, STPT, XC, END                                 PIK 16Ø
      COMMON /GRA/ N, IDPTH, IDEG                                         PIK 17Ø
C IT IS ASSUMED THAT THE GRAPH HAS AT MOST 5Ø COMPONENTS AND              PIK 18Ø
C THAT THERE ARE AT MOST 1ØØ LEVELS.                                      PIK 19Ø
```

```
      COMMON /LVLW/ NHIGH(1ØØ), NLOW(1ØØ), NACUM(1ØØ)                    PIK  2ØØ
      COMMON /CC/ XC, SIZE(5Ø), STPT(5Ø)                                 PIK  21Ø
      DIMENSION LVLS1(1), LVLS2(1), CCSTOR(1)                            PIK  22Ø
C FOR EACH CONNECTED COMPONENT DO                                        PIK  23Ø
      DO 8Ø I=1,XC                                                       PIK  24Ø
        J = STPT(I)                                                      PIK  25Ø
        END = SIZE(I) + J - 1                                            PIK  26Ø
C SET NHIGH AND NLOW EQUAL TO NACUM                                      PIK  27Ø
        DO 1Ø K=1,IDPTH                                                  PIK  28Ø
          NHIGH(K) = NACUM(K)                                            PIK  29Ø
          NLOW(K) = NACUM(K)                                             PIK  3ØØ
   1Ø   CONTINUE                                                         PIK  31Ø
C UPDATE NHIGH AND NLOW FOR EACH NODE IN CONNECTED COMPONENT             PIK  32Ø
        DO 2Ø K=J,END                                                    PIK  33Ø
          INODE = CCSTOR(K)                                              PIK  34Ø
          LVLNH = LVLS1(INODE)                                           PIK  35Ø
          NHIGH(LVLNH) = NHIGH(LVLNH) + 1                                PIK  36Ø
          LVLNL = LVLS2(INODE)                                           PIK  37Ø
          NLOW(LVLNL) = NLOW(LVLNL) + 1                                  PIK  38Ø
   2Ø   CONTINUE                                                         PIK  39Ø
        MAX1 = Ø                                                         PIK  4ØØ
        MAX2 = Ø                                                         PIK  41Ø
C SET MAX1=LARGEST NEW NUMBER IN NHIGH                                   PIK  42Ø
C SET MAX2=LARGEST NEW NUMBER IN NLOW                                    PIK  43Ø
        DO 3Ø K=1,IDPTH                                                  PIK  44Ø
          IF (2*NACUM(K).EQ.NLOW(K)+NHIGH(K)) GO TO 3Ø                   PIK  45Ø
          IF (NHIGH(K).GT.MAX1) MAX1 = NHIGH(K)                          PIK  46Ø
          IF (NLOW(K).GT.MAX2) MAX2 = NLOW(K)                            PIK  47Ø
   3Ø   CONTINUE                                                         PIK  48Ø
C SET IT= NUMBER OF LEVEL STRUCTURE TO BE USED                           PIK  49Ø
        IT = 1                                                           PIK  5ØØ
        IF (MAX1.GT.MAX2) IT = 2                                         PIK  51Ø
        IF (MAX1.EQ.MAX2) IT = IDFLT                                     PIK  52Ø
        IF (IT.EQ.2) GO TO 6Ø                                            PIK  53Ø
        IF (I.EQ.1) ISDIR = -1                                           PIK  54Ø
C COPY LVLS1 INTO LVLS2 FOR EACH NODE IN CONNECTED COMPONENT             PIK  55Ø
        DO 4Ø K=J,END                                                    PIK  56Ø
          INODE = CCSTOR(K)                                              PIK  57Ø
          LVLS2(INODE) = LVLS1(INODE)                                    PIK  58Ø
   4Ø   CONTINUE                                                         PIK  59Ø
C UPDATE NACUM TO BE THE SAME AS NHIGH                                   PIK  6ØØ
        DO 5Ø K=1,IDPTH                                                  PIK  61Ø
          NACUM(K) = NHIGH(K)                                            PIK  62Ø
   5Ø   CONTINUE                                                         PIK  63Ø
        GO TO 8Ø                                                         PIK  64Ø
C UPDATE NACUM TO BE THE SAME AS NLOW                                    PIK  65Ø
   6Ø   DO 7Ø K=1,IDPTH                                                  PIK  66Ø
          NACUM(K) = NLOW(K)                                             PIK  67Ø
   7Ø   CONTINUE                                                         PIK  68Ø
   8Ø CONTINUE                                                           PIK  69Ø
      RETURN                                                             PIK  7ØØ
      END                                                                PIK  71Ø


      SUBROUTINE NUMBER(SND, NUM, NDSTK, LVLS2, NDEG, RENUM, LVLST,      NUM   1Ø
     * LSTPT, NR, NFLG, IBW2, IPF2, IPFA, ISDIR)                         NUM   2Ø
C  NUMBER PRODUCES THE NUMBERING OF THE GRAPH FOR MIN BANDWIDTH          NUM   3Ø
C  SND-          ON INPUT THE NODE TO BEGIN NUMBERING ON                NUM   4Ø
C  NUM-          ON INPUT AND OUTPUT, THE NEXT AVAILABLE NUMBER         NUM   5Ø
C  LVLS2-        THE LEVEL STRUCTURE TO BE USED IN NUMBERING            NUM   6Ø
C  RENUM-        THE ARRAY USED TO STORE THE NEW NUMBERING              NUM   7Ø
C  LVLST-        ON OUTPUT CONTAINS LEVEL STRUCTURE                     NUM   8Ø
C  LSTPT(I)-     ON OUTPUT, INDEX INTO LVLST TO FIRST NODE IN ITH LVL   NUM   9Ø
C                LSTPT(I+1) - LSTPT(I) = NUMBER OF NODES IN ITH LVL     NUM  1ØØ
C  NFLG-         =+1 IF SND IS FORWARD END OF PSEUDO-DIAM               NUM  11Ø
C                =-1 IF SND IS REVERSE END OF PSEUDO-DIAM               NUM  12Ø
C  IBW2-         BANDWIDTH OF NEW NUMBERING COMPUTED BY NUMBER          NUM  13Ø
C  IPF2-         PROFILE OF NEW NUMBERING COMPUTED BY NUMBER            NUM  14Ø
C  IPFA-         WORKING STORAGE USED TO COMPUTE PROFILE AND BANDWIDTH  NUM  15Ø
C  ISDIR-        INDICATES STEP DIRECTION USED IN NUMBERING(+1 OR -1)   NUM  16Ø
C USE INTEGER*2 NDSTK  WITH AN IBM 36Ø OR 37Ø.                           NUM  17Ø
      INTEGER NDSTK                                                      NUM  18Ø
      INTEGER SND, STKA, STKB, STKC, STKD, XA, XB, XC, XD, CX, END,      NUM  19Ø
     * RENUM, TEST                                                       NUM  2ØØ
      COMMON /GRA/ N, IDPTH, IDEG                                        NUM  21Ø
C THE STORAGE IN COMMON BLOCKS CC AND LVLW IS NOW FREE AND CAN            NUM  22Ø
```

```
C BE USED FOR STACKS.                                                  NUM  23Ø
      COMMON /LVLW/ STKA(1ØØ), STKB(1ØØ), STKC(1ØØ)                    NUM  24Ø
      COMMON /CC/ STKD(1ØØ)                                            NUM  25Ø
      DIMENSION IPFA(1)                                                NUM  26Ø
      DIMENSION NDSTK(NR,1), LVLS2(1), NDEG(1), RENUM(1), LVLST(1),    NUM  27Ø
     * LSTPT(1)                                                        NUM  28Ø
C SET UP LVLST AND LSTPT FROM LVLS2                                    NUM  29Ø
      DO 1Ø I=1,N                                                      NUM  3ØØ
        IPFA(I) = Ø                                                    NUM  31Ø
   1Ø CONTINUE                                                         NUM  32Ø
      NSTPT = 1                                                        NUM  33Ø
      DO 3Ø I=1,IDPTH                                                  NUM  34Ø
        LSTPT(I) = NSTPT                                               NUM  35Ø
        DO 2Ø J=1,N                                                    NUM  36Ø
          IF (LVLS2(J).NE.I) GO TO 2Ø                                  NUM  37Ø
          LVLST(NSTPT) = J                                             NUM  38Ø
          NSTPT = NSTPT + 1                                            NUM  39Ø
   2Ø   CONTINUE                                                       NUM  4ØØ
   3Ø CONTINUE                                                         NUM  41Ø
      LSTPT(IDPTH+1) = NSTPT                                           NUM  42Ø
C STKA, STKB, STKC AND STKD ARE STACKS WITH POINTERS                   NUM  43Ø
C XA,XB,XC, AND XD.  CX IS A SPECIAL POINTER INTO STKC WHICH           NUM  44Ø
C INDICATES THE PARTICULAR NODE BEING PROCESSED.                       NUM  45Ø
C LVLN KEEPS TRACK OF THE LEVEL WE ARE WORKING AT.                     NUM  46Ø
C INITIALLY STKC CONTAINS ONLY THE INITIAL NODE, SND.                  NUM  47Ø
      LVLN = Ø                                                         NUM  48Ø
      IF (NFLG.LT.Ø) LVLN = IDPTH + 1                                  NUM  49Ø
      XC = 1                                                           NUM  5ØØ
      STKC(XC) = SND                                                   NUM  51Ø
   4Ø CX = 1                                                           NUM  52Ø
      XD = Ø                                                           NUM  53Ø
      LVLN = LVLN + NFLG                                               NUM  54Ø
      LST = LSTPT(LVLN)                                                NUM  55Ø
      LND = LSTPT(LVLN+1) - 1                                          NUM  56Ø
C BEGIN PROCESSING NODE STKC(CX)                                       NUM  57Ø
   5Ø IPRO = STKC(CX)                                                  NUM  58Ø
      RENUM(IPRO) = NUM                                                NUM  59Ø
      NUM = NUM + ISDIR                                                NUM  6ØØ
      END = NDEG(IPRO)                                                 NUM  61Ø
      XA = Ø                                                           NUM  62Ø
      XB = Ø                                                           NUM  63Ø
C CHECK ALL ADJACENT NODES                                             NUM  64Ø
      DO 8Ø I=1,END                                                    NUM  65Ø
        TEST = NDSTK(IPRO,I)                                           NUM  66Ø
        INX = RENUM(TEST)                                              NUM  67Ø
C ONLY NODES NOT NUMBERED OR ALREADY ON A STACK ARE ADDED              NUM  68Ø
        IF (INX.EQ.Ø) GO TO 6Ø                                         NUM  69Ø
        IF (INX.LT.Ø) GO TO 8Ø                                         NUM  7ØØ
C DO PRELIMINARY BANDWIDTH AND PROFILE CALCULATIONS                    NUM  71Ø
        NBW = (RENUM(IPRO)-INX)*ISDIR                                  NUM  72Ø
        IF (ISDIR.GT.Ø) INX = RENUM(IPRO)                              NUM  73Ø
        IF (IPFA(INX).LT.NBW) IPFA(INX) = NBW                          NUM  74Ø
        GO TO 8Ø                                                       NUM  75Ø
   6Ø   RENUM(TEST) = -1                                               NUM  76Ø
C PUT NODES ON SAME LEVEL ON STKA, ALL OTHERS ON STKB                  NUM  77Ø
        IF (LVLS2(TEST).EQ.LVLS2(IPRO)) GO TO 7Ø                       NUM  78Ø
        XB = XB + 1                                                    NUM  79Ø
        STKB(XB) = TEST                                                NUM  8ØØ
        GO TO 8Ø                                                       NUM  81Ø
   7Ø   XA = XA + 1                                                    NUM  82Ø
        STKA(XA) = TEST                                                NUM  83Ø
   8Ø CONTINUE                                                         NUM  84Ø
C SORT STKA AND STKB INTO INCREASING DEGREE AND ADD STKA TO STKC       NUM  85Ø
C AND STKB TO STKD                                                     NUM  86Ø
      IF (XA.EQ.Ø) GO TO 1ØØ                                           NUM  87Ø
      IF (XA.EQ.1) GO TO 9Ø                                            NUM  88Ø
      CALL SORTDG(STKC, STKA, XC, XA, NDEG)                            NUM  89Ø
      GO TO 1ØØ                                                        NUM  9ØØ
   9Ø XC = XC + 1                                                      NUM  91Ø
      STKC(XC) = STKA(XA)                                              NUM  92Ø
  1ØØ IF (XB.EQ.Ø) GO TO 12Ø                                           NUM  93Ø
      IF (XB.EQ.1) GO TO 11Ø                                           NUM  94Ø
      CALL SORTDG(STKD, STKB, XD, XB, NDEG)                            NUM  95Ø
      GO TO 12Ø                                                        NUM  96Ø
  11Ø XD = XD + 1                                                      NUM  97Ø
      STKD(XD) = STKB(XB)                                              NUM  98Ø
```

```
C BE SURE TO PROCESS ALL NODES IN STKC                                  NUM  99Ø
  12Ø CX = CX + 1                                                       NUM 1ØØØ
      IF (XC.GE.CX) GO TO 5Ø                                            NUM 1Ø1Ø
C WHEN STKC IS EXHAUSTED LOOK FOR MIN DEGREE NODE IN SAME LEVEL         NUM 1Ø2Ø
C WHICH HAS NOT BEEN PROCESSED                                          NUM 1Ø3Ø
      MAX = IDEG + 1                                                    NUM 1Ø4Ø
      SND = N + 1                                                       NUM 1Ø5Ø
      DO 13Ø I=LST,LND                                                  NUM 1Ø6Ø
        TEST = LVLST(I)                                                 NUM 1Ø7Ø
        IF (RENUM(TEST).NE.Ø) GO TO 13Ø                                 NUM 1Ø8Ø
        IF (NDEG(TEST).GE.MAX) GO TO 13Ø                                NUM 1Ø9Ø
        RENUM(SND) = Ø                                                  NUM 11ØØ
        RENUM(TEST) = -1                                                NUM 111Ø
        MAX = NDEG(TEST)                                                NUM 112Ø
        SND = TEST                                                      NUM 113Ø
  13Ø CONTINUE                                                          NUM 114Ø
      IF (SND.EQ.N+1) GO TO 14Ø                                         NUM 115Ø
      XC = XC + 1                                                       NUM 116Ø
      STKC(XC) = SND                                                    NUM 117Ø
      GO TO 5Ø                                                          NUM 118Ø
C IF STKD IS EMPTY WE ARE DONE, OTHERWISE COPY STKD ONTO STKC           NUM 119Ø
C AND BEGIN PROCESSING NEW STKC                                         NUM 12ØØ
  14Ø IF (XD.EQ.Ø) GO TO 16Ø                                            NUM 121Ø
      DO 15Ø I=1,XD                                                     NUM 122Ø
        STKC(I) = STKD(I)                                               NUM 123Ø
  15Ø CONTINUE                                                          NUM 124Ø
      XC = XD                                                           NUM 125Ø
      GO TO 4Ø                                                          NUM 126Ø
C DO FINAL BANDWIDTH AND PROFILE CALCULATIONS                           NUM 127Ø
  16Ø DO 17Ø I=1,N                                                      NUM 128Ø
        IF (IPFA(I).GT.IBW2) IBW2 = IPFA(I)                             NUM 129Ø
        IPF2 = IPF2 + IPFA(I)                                           NUM 13ØØ
  17Ø CONTINUE                                                          NUM 131Ø
      RETURN                                                            NUM 132Ø
      END                                                               NUM 133Ø
```


REMARK ON ALGORITHMS 508 AND 509

Matrix Bandwidth and Profile Reduction [H.L. Crane, Jr., N.E. Gibbs, W.G. Poole, Jr., and P.K. Stockmeyer, *ACM Trans. Math. Softw. 2*, 4 (Dec. 1976), 375–377] and A Hybrid Profile Reduction Algorithm [N.E. Gibbs, *ACM Trans. Math. Softw. 2*, 4 (Dec. 1976), 378–387]

John G. Lewis [Received 28 May 1980; revised 6 January 1982; accepted 10 January 1982]

Boeing Computer Services Co., Mail Stop 9C-01, 565 Andover Park West, Tukwila, WA 98188.

A new implementation of the Gibbs–Poole–Stockmeyer and Gibbs–King algorithms is available as Algorithm 582. This new implementation is faster, more robust, and requires less storage than the previous implementation of the Gibbs–Poole–Stockmeyer algorithm [1], distributed as Algorithm 508, and of the Gibbs–King algorithm [2], distributed as Algorithm 509. The mathematical capabilities of Algorithm 582 are identical to those of Algorithms 508 and 509. References [3] and [4] give the implementation details and documentation for the new implementation.

REFERENCES

1. CRANE, H.L., JR., GIBBS, N.E., POOLE, W. G., JR., AND STOCKMEYER, P.K. Matrix bandwidth and profile reduction. *ACM Trans. Math. Softw. 2*, 4 (Dec. 1976), 375–377.
2. GIBBS, N.E. A hybrid profile reduction algorithm. *ACM Trans. Math. Softw. 2*, 4 (Dec. 1976), 378–387.
3. LEWIS, J.G. Implementation of the Gibbs–Poole–Stockmeyer and Gibbs–King algorithms. *ACM Trans. Math. Softw. 8*, 2 (June 1982), 180–189.
4. LEWIS, J.G. Algorithm 582. The Gibbs–Poole–Stockmeyer and Gibbs–King algorithms for reordering Sparse matrices. *ACM Trans. Math. Softw. 8*, 2 (June 1982), 190–194.

ALGORITHM 509
A Hybrid Profile Reduction Algorithm [F1]

NORMAN E. GIBBS
College of William and Mary

Key Words and Phrases: bandwidth reduction, graph, King algorithm, leveling, profile reduction, sparse matrix
CR Categories: 5.14, 5.32
Language: Fortran

DESCRIPTION

1. Introduction

Large sparse matrix problems arise in many applications areas, such as structural engineering, fluid dynamics, and network analysis. Many algorithms for automatically reordering the rows and columns of the matrix in order to reduce bandwidth and/or profile have been published. Recently, Gibbs et al. [4] concluded that of the many reduction algorithms available, theirs (as described in [3] and implemented as a Fortran program in [1]) and King's (as described in [5] and implemented as a Fortran program as described in [4]) were superior for profile reduction. Although the King algorithm produced the best profile in many cases, it sometimes exhibited erratic behavior due to the strategy of reducing the profile globally by reducing the profile well locally. Furthermore, they found a large class of problems (wide cylinders—Type E3 in [4]) for which there is no good single starting point for the King algorithm.

In this paper a new "hybrid" algorithm for reducing profile is presented. This algorithm combines the better features of [3] and [5] by first finding a pseudodiameter to produce a leveling and then numbering the leveling level by level according to King's criteria.

Section 2 describes the new profile reduction algorithm. In Section 3 it is compared in production mode with the better algorithms reported on in [4]. A listing of the Fortran subprogram used to implement the new algorithm is given in Appendix A. The changes which must be made to the Fortran subroutine REDUCE [1] to implement the algorithm are given in Appendix B.

2. Description of the Algorithm

Let $Ax = b$ be an $n \times n$ sparse nonsingular system of linear algebraic equations. To define the profile of A, first define $f_i = \min \{j : a_{ij} \neq 0\}$ for $i = 1, 2, \ldots, n$ (it is assumed that $a_{ii} \neq 0$). This locates the leftmost nonzero element in each row.

Received 31 August 1975 and 3 March 1976.

This research was supported by the Office of Naval Research under Contract N00014-73-A-374-0001, NR-44-459.
Author's address: Department of Mathematics, College of William and Mary, Williamsburg, VA 23185.

Now set $\delta_i = i - f_i$. The *profile* is defined to be $\sum_{i=1}^{n} \delta_i$. Given such a matrix A, we define a *graph* $G = \langle V, E \rangle$ where V has n *vertices*, $\{v_1, v_2, \ldots, v_n\}$ and $\{v_i, v_j\} \in E$ if $a_{ij} \neq 0$ and $i \neq j$. It is well known (see, for example [3]) that the problem of matrix profile reduction can be restated as a problem of graph profile reduction. The algorithm presented here is described in terms of graphs and is directly applicable to graphs.

We assume that the reader is familiar with Algorithms I and II of Gibbs et al. [3] and with King's algorithm [5]. The reader is referred to [3] for the definitions and notation used below.

Algorithms I and II accept as input a connected graph and produce as output a leveling. A *leveling* of a graph $G = \langle V, E \rangle$ is a partition of the vertices V into levels $L_1, L_2, \ldots, L_k$ such that

(1) all vertices adjacent to vertices in level L_1 are either in level L_1 or level L_2;
(2) all vertices adjacent to vertices in level L_k are either in level L_{k-1} or level L_k;
(3) for $1 < i < k$, all vertices adjacent to vertices in level L_i are in either level L_{i-1}, level L_i, or level L_{i+1}.

The new algorithm, Algorithm P, described below is applied to the leveling of a connected component of a graph. The following assumptions are made.

(1) $L_1, L_2, \ldots, L_k$ is a leveling produced by algorithms I and II and the number of levels, k, must be at least two.
(2) On input the variable *nextnum* is initialized to the next available integer to be assigned to this connected component. On output *nextnum* will be set to the next available integer to be assigned to the next connected component.
(3) The variables $S2$, $S3$, and Q represent queues of integers. If X is a queue of integers and x is an integer variable, then:
    (a) $X \leftarrow X - \{x\}$ means remove the integer x from the queue X (treating X as a set).
    (b) $X \Leftarrow x$ means place the integer x at the rear of the queue X.
    (c) $x \Leftarrow X$ means set the variable x to the integer at the front of the queue X after removing it from X.
    (d) $X \leftarrow \emptyset$ means empty the queue X.
    (e) $| X |$ represents the current size of the queue X.
(4) Algorithm P produces as output a vector *new*, where $new(i) = j$ means that the old vertex labeled with the integer i should be relabeled with the integer j.

ALGORITHM P. PROFILE REDUCTION

P0. [Initialization.] Set $S2 \leftarrow L_1$ and $i \leftarrow 1$.
P1. [Form the next level and empty Q.] Set $S3 \leftarrow L_{i+1}$ and $Q \leftarrow \emptyset$.
P2. [Find the next vertex to be numbered according to King's criteria.] Set m equal to the first element of $S2$ which has fewest connections to $S3$. Set C equal to the set of elements of $S3$ which are joined by an edge to m.
P3. [Numbering.] Set $new(m) \leftarrow nextnum$, $S2 \leftarrow S2 - \{m\}$, $nextnum \leftarrow nextnum + 1$, and for each element $c \in C$ set $Q \Leftarrow c$ and $S3 \leftarrow S3 - \{c\}$.
P4. [Is $S2$ empty?] If $| S2 | = 0$ goto step P7.
P5. [$S2$ is not empty. Is $S3$ empty?] If $| S3 | > 0$ return to step P2.
P6. [$S2$ is not empty, but $S3$ is empty.] For $j = 1, 2, \ldots, | S2 |$ set $s \Leftarrow S2$, $new(s) \leftarrow nextnum$, and $nextnum \leftarrow nextnum + 1$. Goto step P8.
P7. [$S2$ is empty. If $S3$ is not empty place the rest of S3 in Q.] If $| S3 | > 0$ then set $q \Leftarrow S3$, $Q \Leftarrow q$, and repeat step P7.
P8. [$S2$ and $S3$ are empty.] Set $i \leftarrow i + 1$. If $i < k$ (the last level) then set $S2 \leftarrow Q$ and return to step P1.
P9. [All levels are numbered except the last level which is in Q.] If $| Q | = 0$ then terminate the algorithm; otherwise set $q \Leftarrow Q$, $new(q) \leftarrow nextnum$, $nextnum \leftarrow nextnum + 1$, and repeat step P9.

Algorithm P assigns consecutive integers, level by level, to the leveling according to King's criteria. In fact, this algorithm is the King algorithm restricted to assigning consecutive integers to the vertices in one level at a time. If initially $S2$ was a single vertex v, $S3$ was set to $V - \{v\}$, and the elements of C were placed directly into $S2$ instead of Q in step P3 we would, in essence, have King's algorithm.

It turns out that reversing the numbering *new* by setting $new(j) \leftarrow n - new(j) + 1$ for $j = 1, 2, \ldots, n$ sometimes improves profile. We have not been able to predict a priori when reversing helps, so every numbering is reversed and the numbering that gives smaller profile is chosen (see subroutine CHECK in Appendix A.)

3. Test Results in Production Mode

The new algorithm as described above and implemented as a Fortran subroutine named PROFIT (see Appendix A) was integrated into the Fortran package described in [1]. A parameter was added to subroutine REDUCE so that the user can specify whether profile reduction is more important than bandwidth reduction or if bandwidth reduction is more important than profile reduction. The changes which were made to REDUCE are described in Appendix B.

As stated in [3], every practical bandwidth or profile reduction algorithm is heuristic in the sense that one cannot make absolute a priori statements concerning the performance of the algorithm; the performance is data dependent. In order to relate this work to previous work the new algorithm was tested on the same test data described in [4]. Tables I and II summarize the results. These tables show how subroutine REDUCE with subroutine PROFIT performed relative to Gibbs et al. (GPS [1, 3]) and King (KNG [5, 4]), with respect to bandwidth reduction, profile reduction, and time. The times are relative to 1.00 for GPS.

4. Conclusions

Although the new algorithm takes more time than GPS, it is clear that it is superior to GPS in reducing profile. The new algorithm yields profile comparable to KNG, but takes less time and does not exhibit KNG's erratic behavior.

In retrospect, the good performance of GPS for bandwidth reduction and the new algorithm for profile reduction can be explained. The Cuthill-McKee [2] and King algorithms do a good job of reducing bandwidth and profile respectively by numbering well locally. Since these algorithms examine the graph's structure locally, they sometimes behave erratically. The algorithms for producing a leveling by first finding a pseudodiameter and then using it to minimize level width take the entire structure of the graph into account and thus, when the Cuthill-McKee or King numbering is imposed on a leveling one obtains comparable bandwidth or profile respectively. The GPS algorithm and the new algorithm give superior bandwidth or profile reduction on those cases where global strategy is necessary.

Table I. Applications Problems
See [3] for a description.
n: order of matrix; β: bandwith; ρ: profile; RATIO: ratio to GPS time

		Original		GPS		Profit			KNG		
	n	β	ρ	β	ρ	β	ρ	Ratio	β	ρ	Ratio
1	68	45	598	7	269	8	270	1.41	30	216	21.48
2	90	85	1020	7	579	7	579	1.52	11	573	5.66
3	92	80	2127	13	736	17	725	2.69	38	673	13.16
4	130	126	3615	18	1588	21	1482	3.43	25	1496	5.17
5	159	19	1046	12	971	15	964	1.75	19	1046	3.68
6	174	16	1569	13	1466	16	1405	2.28	28	1341	5.30
7	185	168	7534	29	3610	47	3485	6.81	116	2882	11.78
8	220	166	8532	12	1868	12	1791	2.01	13	1754	92.62
9	263	262	2681	19	2346	21	2265	2.20	28	2152	33.32
10	263	30	2040	14	2001	18	1932	1.97	30	1823	13.71
11	310	302	23357	14	2726	21	2697	2.10	25	2660	18.90
12	312	262	18076	37	5548	51	5238	7.71	190	6679	8.36
13	346	216	16435	46	7650	55	7334	10.79	66	6302	20.20
14	360	344	29790	34	6364	52	5800	6.61	316	8875	9.34
15	436	173	7913	33	7844	47	7550	4.99	173	7913	6.20
16	512	399	35837	29	4669	29	4304	1.78	59	4134	2.96
17	555	480	56322	91	28976	117	27247	59.50	179	23106	28.03
18	861	833	100560	71	45525	115	41904	37.66	436	54159	44.34
19	918	840	124607	49	20369	64	19573	5.87	737	54104	17.24

Table II. Selected Grid Problems

See [4] for a complete description of grids. * indicates test not run.

n: order of matrix; β: bandwidth; ρ: profile; RATIO: ratio to GPS time

A1: square grid (5-point difference scheme); B3: 3×1 rectangle (9-point difference scheme)

D3: 20×1 cylinder (9-point difference scheme); E2: 1×20 cylinder (7-point difference scheme)

		Original		GPS		Profit			KNG		
	n	β	ρ	β	ρ	β	ρ	Ratio	β	ρ	Ratio
A1	49	47	406	7	245	7	245	1.750	7	245	27.33
A1	100	98	1153	10	705	10	705	2.111	11	705	51.57
A1	400	398	8713	20	5510	20	5510	4.068	21	5510	156.28
A1	625	623	16768	25	10700	25	10700	5.253	25	10700	158.45
B3	48	46	366	5	212	5	212	1.267	7	211	4.68
B3	432	430	7078	13	5436	13	5436	2.389	23	5371	7.02
B3	1200	1198	29406	21	24740	21	24740	3.987	39	24503	7.86
B3	1728	1726	49306	25	42552	25	42552	4.801	47	42209	8.27
D3	500	498	71474	7	3175	7	3175	1.379	9	3172	9.64
D3	980	978	218512	9	8351	9	8351	1.575	13	8341	13.19
D3	1620	1618	490830	11	17199	11	17199	1.767	17	17178	18.65
D3	2420	2418	926828	13	30679	13	30679	2.005	21	30643	22.96
E2	320	318	3367	10	2662	9	2512	2.071	13	2507	93.76
E2	720	718	10541	14	8942	14	8486	3.040	23	8468	117.30
E2	1280	1278	23931	18	21038	19	20120	4.197	*	*	*
E2	2000	1998	45457	22	40838	24	39302	5.174	*	*	*

REFERENCES

1. CRANE, H.L., GIBBS, N.E., POOLE, W.G., AND STOCKMEYER, P.K. Algorithm 508. Matrix bandwidth and profile reduction. *ACM Trans. Math. Software 2*, 4 (Dec. 1976), 375–377. (this issue)
2. CUTHILL, E.H., AND MCKEE, J. Reducing the bandwidth of sparse symmetric matrices. Proc. 24th ACM Nat. Conf., ACM, New York, 1969, pp. 157–172.
3. GIBBS, N.E., POOLE, W.G., AND STOCKMEYER, P.K. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM J. Numer. Analysis 13*, 2 (April 1976), 235–251.
4. GIBBS, N.E., POOLE, W.G., AND STOCKMEYER, P.K. A comparison of several bandwidth and profile reduction algorithms. *ACM Trans. Math. Software 2*, 4 (Dec. 1976), 322–330. (this issue)
5. KING, I.P. An automatic reordering scheme for simultaneous equations derived from network analysis. Int. J. Numer. Methods Engrg. *2* (1970), 523–533.

APPENDIX A

```
      SUBROUTINE PROFIT(NR, NDSTK, NEW, NDEG, LVLS2, LVLST, LSTPT,      PRO   1Ø
     * NXTNUM)                                                          PRO   2Ø
C SUBROUTINE PROFIT NUMBERS LEVEL BY LEVEL WITH CONSECUTIVE INTEGERS    PRO   3Ø
C USING A MODIFIED VERSION OF KING*S ALGORITHM.                         PRO   4Ø
C   NR-         ROW DIMENSION OF CONNECTION TABLE.                      PRO   5Ø
C   NDSTK-      THE CONNECTION TABLE.                                   PRO   6Ø
C   NEW-        VECTOR TO STORE THE NEW NUMBERING.                      PRO   7Ø
C   NDEG(I)-    THE DEGREE OF NODE I.                                   PRO   8Ø
C   LVLS2-      THE LEVEL STRUCTURE PRODUCED BY PIKLVL.                 PRO   9Ø
C               LVLS2(I) = J MEANS VERTEX I HAS BEEN                    PRO  1ØØ
C               PLACED IN LEVEL J.                                      PRO  11Ø
C   LVLST-      ON OUTPUT,  CONTAINS THE LEVEL STRUCTURE USED.          PRO  12Ø
C               LVLST(LSTPT(I)),...,LVLST(LSTPT(I+1)-1) ARE             PRO  13Ø
C               THE VERTICES IN LEVEL I.                                PRO  14Ø
C   LSTPT(I)-   ON OUTPUT, INDEX INTO LVLST TO FIRST NODE IN LEVEL I.   PRO  15Ø
C               LSTPT(I+1) - LSTPT(I) = NUMBER OF NODES IN I*TH LEVEL.  PRO  16Ø
C   NXTNUM-     ON INPUT AND OUTPUT,  THE NEXT AVAILABLE NUMBER.        PRO  17Ø
C ON IBM 36Ø OR 37Ø USE INTEGER * 2 NDSTK.                              PRO  18Ø
      INTEGER NDSTK                                                     PRO  19Ø
      INTEGER NEW(1), NDEG(1), LVLS2(1), LVLST(1), LSTPT(1)             PRO  2ØØ
      DIMENSION NDSTK(NR,1)                                             PRO  21Ø
C COMMON AREA GRA HOLDS VITAL INFORMATION ABOUT THE GRAPH                PRO  22Ø
C N-          THE NUMBER OF NODES                                       PRO  23Ø
C IDPTH-      THE NUMBER OF LEVELS FOUND BY PIKLVL.                     PRO  24Ø
C IDEG-       MAXIMUM DEGREE OF GRAPH -- COLUMN DIMENSION OF NDSTK.     PRO  25Ø
      COMMON /GRA/ N, IDPTH, IDEG                                       PRO  26Ø
C IT IS ASSUMED THAT NO LEVEL HAS MORE THAN 1ØØ NODES.                  PRO  27Ø
      COMMON /LVLW/ S2(1ØØ), S3(1ØØ), Q(1ØØ)                           PRO  28Ø
      COMMON /CC/ CONECT(1ØØ)                                           PRO  29Ø
      INTEGER S2, S3, Q, CONECT, S2SZE, S3SZE, QPTR, CONSZE             PRO  3ØØ
C SET UP LVLST AND LSTPT FROM LVLS2.                                    PRO  31Ø
      NSTPT = 1                                                         PRO  32Ø
      DO 2Ø I=1,IDPTH                                                   PRO  33Ø
        LSTPT(I) = NSTPT                                                PRO  34Ø
        DO 1Ø J=1,N                                                     PRO  35Ø
          IF (LVLS2(J).NE.I) GO TO 1Ø                                   PRO  36Ø
          LVLST(NSTPT) = J                                              PRO  37Ø
          NSTPT = NSTPT + 1                                             PRO  38Ø
   1Ø   CONTINUE                                                        PRO  39Ø
   2Ø CONTINUE                                                          PRO  4ØØ
      LSTPT(IDPTH+1) = NSTPT                                            PRO  41Ø
C ******************  STEP PØ   ****************************************PRO  42Ø
C  S2 IS THE FIRST LEVEL.                                               PRO  43Ø
      LEVEL = 1                                                         PRO  44Ø
      CALL FORMLV(S2, S2SZE, LSTPT, LVLST, LEVEL)                       PRO  45Ø
C ******************  STEP P1   ****************************************PRO  46Ø
C  S3 IS THE LEVEL ADJACENT TO THE LEVEL S2.                            PRO  47Ø
C  Q IS A QUEUE USED TO RETAIN THE ORDER IN WHICH THE ELEMENTS OF S3    PRO  48Ø
C  ARE REMOVED.  Q EVENTUALLY BECOMES THE NEW S2 AND IS ORDERED         PRO  49Ø
C  ACCORDING TO KING*S CRITERA.                                         PRO  5ØØ
   3Ø CALL FORMLV(S3, S3SZE, LSTPT, LVLST, LEVEL+1)                     PRO  51Ø
      QPTR = Ø                                                          PRO  52Ø
C ******************  STEP P2   ****************************************PRO  53Ø
C  FIND THE NODE M IN S2 WHICH IS ADJACENT TO THE FEWEST NODES IN S3.   PRO  54Ø
   4Ø M = MINCON(S2,S2SZE,S3,S3SZE,CONECT,CONSZE,NDSTK,NR,NDEG)         PRO  55Ø
C ******************  STEP P3   ****************************************PRO  56Ø
C  NUMBER M AND REMOVE IT FROM S2.                                      PRO  57Ø
      NEW(M) = NXTNUM                                                   PRO  58Ø
      NXTNUM = NXTNUM + 1                                               PRO  59Ø
```

```
      CALL DELETE(S2, S2SZE, M)                                          PRO  6ØØ
      IF (CONSZE.LE.Ø) GO TO 6Ø                                          PRO  61Ø
C THE ELEMENTS OF CONLST ARE TO BE REMOVED FROM S3 AND PLACED INTO       PRO  62Ø
C Q.                                                                     PRO  63Ø
      DO 5Ø I=1,CONSZE                                                   PRO  64Ø
        QPTR = QPTR + 1                                                  PRO  65Ø
        Q(QPTR) = CONECT(I)                                              PRO  66Ø
        CALL DELETE(S3, S3SZE, CONECT(I))                                PRO  67Ø
   5Ø CONTINUE                                                           PRO  68Ø
C ******************  STEP P4   ****************************************PRO  69Ø
   6Ø IF (S2SZE.LE.Ø) GO TO 8Ø                                           PRO  7ØØ
C ******************  STEP P5   ****************************************PRO  71Ø
      IF (S3SZE.GT.Ø) GO TO 4Ø                                           PRO  72Ø
C ******************  STEP P6   ****************************************PRO  73Ø
C  S3 IS EMPTY, BUT S2 IS NOT.  RENUMBER THE NODES WHICH REMAIN IN S2.   PRO  74Ø
      DO 7Ø I=1,S2SZE                                                    PRO  75Ø
        NS2 = S2(I)                                                      PRO  76Ø
        NEW(NS2) = NXTNUM                                                PRO  77Ø
        NXTNUM = NXTNUM + 1                                              PRO  78Ø
   7Ø CONTINUE                                                           PRO  79Ø
      GO TO 1ØØ                                                          PRO  8ØØ
C ******************  STEP P7   ****************************************PRO  81Ø
   8Ø IF (S3SZE.LE.Ø) GO TO 1ØØ                                          PRO  82Ø
C S2 IS EMPTY, BUT S3 IS NOT.  MOVE S3*S REMAINING NODES INTO Q.         PRO  83Ø
      DO 9Ø I=1,S3SZE                                                    PRO  84Ø
        QPTR = QPTR + 1                                                  PRO  85Ø
        Q(QPTR) = S3(I)                                                  PRO  86Ø
   9Ø CONTINUE                                                           PRO  87Ø
C ******************  STEP P8   ****************************************PRO  88Ø
  1ØØ LEVEL = LEVEL + 1                                                  PRO  89Ø
      IF (LEVEL.GE.IDPTH) GO TO 12Ø                                      PRO  9ØØ
C S2 BECOMES THE OLD Q SINCE BOTH S2 AND S3 ARE EMPTY.                   PRO  91Ø
      DO 11Ø I=1,QPTR                                                    PRO  92Ø
        S2(I) = Q(I)                                                     PRO  93Ø
  11Ø CONTINUE                                                           PRO  94Ø
      S2SZE = QPTR                                                       PRO  95Ø
      GO TO 3Ø                                                           PRO  96Ø
C ******************  STEP P9   ****************************************PRO  97Ø
C  LAST LEVEL IS ORDERED IN Q,  SO NUMBER IT BEFORE RETURNING.           PRO  98Ø
  12Ø DO 13Ø I=1,QPTR                                                    PRO  99Ø
        IQ = Q(I)                                                        PRO 1ØØØ
        NEW(IQ) = NXTNUM                                                 PRO 1Ø1Ø
        NXTNUM = NXTNUM + 1                                              PRO 1Ø2Ø
  13Ø CONTINUE                                                           PRO 1Ø3Ø
      RETURN                                                             PRO 1Ø4Ø
      END                                                                PRO 1Ø5Ø


      FUNCTION MINCON(X, XSZE, Y, YSZE, CONLST, CONSZE, NDSTK, NR,       MIN   1Ø
     * NDEG)                                                             MIN   2Ø
C FUNCTION MINCON RETURNS AS ITS FUNCTIONAL VALUE A VERTEX X(I) SUCH     MIN   3Ø
C THAT THE NUMBER OF CONNECTIONS FROM X(I) TO THE SET Y IS A MINIMUM.    MIN   4Ø
C THE VERTICES OF Y WHICH ARE ADJACENT TO X(I) ARE PLACED IN             MIN   5Ø
C CONLST(1), CONLST(2),...,CONLST(CONSZE).                               MIN   6Ø
C USE INTEGER * 2 NDSTK ON IBM 36Ø OR 37Ø.                               MIN   7Ø
      INTEGER NDSTK                                                      MIN   8Ø
      DIMENSION NDSTK(NR,1)                                              MIN   9Ø
      INTEGER X(1), XSZE, Y(1), YSZE, CONLST(1), CONSZE, NDEG(1)         MIN  1ØØ
C IT IS ASSUMED THAT NO LEVEL HAS MORE THAN 1ØØ VERTICES.                MIN  11Ø
      INTEGER SMLST(1ØØ)                                                 MIN  12Ø
      CONSZE = YSZE + 1                                                  MIN  13Ø
      DO 5Ø I=1,XSZE                                                     MIN  14Ø
        LSTSZE = Ø                                                       MIN  15Ø
        IX = X(I)                                                        MIN  16Ø
        IROWDG = NDEG(IX)                                                MIN  17Ø
        DO 2Ø J=1,YSZE                                                   MIN  18Ø
          DO 1Ø K=1,IROWDG                                               MIN  19Ø
            IX = X(I)                                                    MIN  2ØØ
            IF (NDSTK(IX,K).NE.Y(J)) GO TO 1Ø                            MIN  21Ø
            SMLST(LSTSZE+1) = Y(J)                                       MIN  22Ø
            LSTSZE = LSTSZE + 1                                          MIN  23Ø
            IF (LSTSZE.GE.CONSZE) GO TO 5Ø                               MIN  24Ø
            GO TO 2Ø                                                     MIN  25Ø
   1Ø     CONTINUE                                                       MIN  26Ø
   2Ø   CONTINUE                                                         MIN  27Ø
        IF (LSTSZE.GT.Ø) GO TO 3Ø                                        MIN  28Ø
```

```
C WE HAVE FOUND A VERTEX IN X WHICH IS NOT CONNECTED TO ANY VERTEX      MIN  29Ø
C IN Y                                                                 MIN  3ØØ
        MINCON = X(I)                                                  MIN  31Ø
        CONSZE = Ø                                                     MIN  32Ø
        RETURN                                                         MIN  33Ø
C WE HAVE FOUND A VERTEX X(I) WITH FEWEST CONNECTIONS (NONZERO) TO Y   MIN  34Ø
C SO FAR.  SAVE THE ELEMENTS OF Y WHICH CONNECT TO X(I) IN CONLST AND  MIN  35Ø
C SAVE X(I) AS THE FUNCTIONAL VALUE.                                   MIN  36Ø
   3Ø   CONSZE = LSTSZE                                                MIN  37Ø
        DO 4Ø J=1,LSTSZE                                               MIN  38Ø
          CONLST(J) = SMLST(J)                                         MIN  39Ø
   4Ø   CONTINUE                                                       MIN  4ØØ
        MINCON = X(I)                                                  MIN  41Ø
   5Ø CONTINUE                                                         MIN  42Ø
      RETURN                                                           MIN  43Ø
      END                                                              MIN  44Ø


      SUBROUTINE DELETE(SET, SETSZE, ELEMNT)                           DEL   1Ø
C SUBROUTINE DELETE REMOVES ELEMNT FROM THE SET SET IF ELEMNT          DEL   2Ø
C IS IN SET.  OTHERWISE,  IT ISSUES A DIAGNOSTIC.                      DEL   3Ø
      INTEGER SET(1), SETSZE, ELEMNT                                   DEL   4Ø
      IF (SETSZE.GT.1) GO TO 1Ø                                        DEL   5Ø
      IF (SETSZE.EQ.1 .AND. SET(1).NE.ELEMNT) GO TO 3Ø                 DEL   6Ø
      SETSZE = Ø                                                       DEL   7Ø
      RETURN                                                           DEL   8Ø
   1Ø DO 2Ø I=1,SETSZE                                                 DEL   9Ø
        IF (SET(I).EQ.ELEMNT) GO TO 4Ø                                 DEL  1ØØ
   2Ø CONTINUE                                                         DEL  11Ø
   3Ø WRITE (6,99999) ELEMNT, (SET(I),I=1,SETSZE)                      DEL  12Ø
      RETURN                                                           DEL  13Ø
   4Ø SETSZE = SETSZE - 1                                              DEL  14Ø
      DO 5Ø J=I,SETSZE                                                 DEL  15Ø
        SET(J) = SET(J+1)                                              DEL  16Ø
   5Ø CONTINUE                                                         DEL  17Ø
      RETURN                                                           DEL  18Ø
99999 FORMAT (1ØHØERROR -- , I6, 8H NOT IN , (2ØI5))                   DEL  19Ø
      END                                                              DEL  2ØØ


      SUBROUTINE FORMLV(SET, SETSZE, LSTPT, LVLST, LEVEL)              FOR   1Ø
C FORMLVL COPIES LEVEL(LEVEL) INTO SET.                                FOR   2Ø
      INTEGER SET(1), SETSZE, LSTPT(1), LVLST(1), UPPER                FOR   3Ø
      LOWER = LSTPT(LEVEL)                                             FOR   4Ø
      UPPER = LSTPT(LEVEL+1) - 1                                       FOR   5Ø
      SETSZE = 1                                                       FOR   6Ø
      DO 1Ø I=LOWER,UPPER                                              FOR   7Ø
        SET(SETSZE) = LVLST(I)                                         FOR   8Ø
        SETSZE = SETSZE + 1                                            FOR   9Ø
   1Ø CONTINUE                                                         FOR  1ØØ
      SETSZE = SETSZE - 1                                              FOR  11Ø
      RETURN                                                           FOR  12Ø
      END                                                              FOR  13Ø


      SUBROUTINE CHECK(BESTBW, BESTPF, RENUM, NDSTK, NR, NDEG, IWK)    CHE   1Ø
C SUBROUTINE CHECK TESTS TO SEE IF REVERSED NUMBERING GIVES BETTER     CHE   2Ø
C PROFILE THAN PROFIT.  IF IT DOES, THEN RENUM IS REVERSED AND BESTPF  CHE   3Ø
C IS SET TO THE SMALLEST OF RENUM AND REVERSED RENUM.                  CHE   4Ø
C USE INTEGER * 2 NDSTK ON IBM 36Ø OR 37Ø                              CHE   5Ø
      INTEGER NDSTK                                                    CHE   6Ø
      DIMENSION NDSTK(NR,1)                                            CHE   7Ø
      INTEGER BESTBW, BESTPF, RENUM(1), NDEG(1), IWK(1)                CHE   8Ø
      COMMON /GRA/ N, IDPTH, IDEG                                      CHE   9Ø
      DO 1Ø I=1,N                                                      CHE  1ØØ
        IWK(I) = N - RENUM(I) + 1                                      CHE  11Ø
   1Ø CONTINUE                                                         CHE  12Ø
      CALL BAND(BESTBW, BESTPF, RENUM, NDSTK, NR, NDEG)                CHE  13Ø
      CALL BAND(IBW, IPF, IWK, NDSTK, NR, NDEG)                        CHE  14Ø
      IF (IPF.GE.BESTPF) RETURN                                        CHE  15Ø
      DO 2Ø I=1,N                                                      CHE  16Ø
        RENUM(I) = IWK(I)                                              CHE  17Ø
   2Ø CONTINUE                                                         CHE  18Ø
      BESTPF = IPF                                                     CHE  19Ø
      RETURN                                                           CHE  2ØØ
      END                                                              CHE  21Ø
```

```
      SUBROUTINE BAND(IBW, IPF, NEW, NDSTK, NR, NDEG)                    BAN  1Ø
C SUBROUTINE BAND COMPUTES THE BANDWIDTH IBW AND THE PROFILE             BAN  2Ø
C IPF OF THE GRAPH REPRESENTED BY NDSTK USING THE NUMBERING NEW.         BAN  3Ø
C ON IBM 36Ø OR 37Ø USE INTEGER * 2 NDSTK.                               BAN  4Ø
      INTEGER NDSTK                                                      BAN  5Ø
      DIMENSION NDSTK(NR,1), NEW(1), NDEG(1)                             BAN  6Ø
      COMMON /GRA/ N, IDPTH, IDEG                                        BAN  7Ø
      IPF = Ø                                                            BAN  8Ø
      IBW = Ø                                                            BAN  9Ø
      DO 2Ø K=1,N                                                        BAN 1ØØ
        IEND = NDEG(K)                                                   BAN 11Ø
        IF (IEND.EQ.Ø) GO TO 2Ø                                          BAN 12Ø
        NBW = Ø                                                          BAN 13Ø
        DO 1Ø J=1,IEND                                                   BAN 14Ø
          IDUMMY = NDSTK(K,J)                                            BAN 15Ø
          NTEST = NEW(K) - NEW(IDUMMY)                                   BAN 16Ø
          IF (NTEST.LE.NBW) GO TO 1Ø                                     BAN 17Ø
          NBW = NTEST                                                    BAN 18Ø
   1Ø   CONTINUE                                                         BAN 19Ø
        IPF = IPF + NBW                                                  BAN 2ØØ
        IF (NBW.GT.IBW) IBW = NBW                                        BAN 21Ø
   2Ø CONTINUE                                                           BAN 22Ø
      RETURN                                                             BAN 23Ø
      END                                                                BAN 24Ø
```

APPENDIX B. SUMMARY OF CHANGES TO FORTRAN SUBROUTINE REDUCE NECESSARY TO IMPLEMENT ALGORITHM P

There are seven changes which have to be made to REDUCE to use the subprograms given in Appendix A. The first is to add the variable OPTPRO to REDUCE's argument list. One change, the sixth, requires adding a statement number. The remaining five changes require inserting statements in the proper place in REDUCE. The update cards are underlined and at least one line of REDUCE before and after each alteration is given.

```
      SUBROUTINE REDUCE(NDSTK, NR, IOLD, RENUM, NDEG, LVL, LVLS1,        RED   10
     * LVLS2, CCSTOR, IBW2, IPF2, OPTPRO)                                RED   20
C  SUBROUTINE REDUCE DETERMINES A ROW AND COLUMN PERMUTATION WHICH,       RED   30
                                  .
                                  .
                                  .
C     IDEG-       MAXIMUM DEGREE OF ANY NODE IN THE GRAPH.               RED  330
C     OPTPRO-     SET TO 1 BY CALLING PROGRAM IF PROFILE REDUCTION        RED  331
C                 IS MORE IMPORTANT THAN BANDWIDTH REDUCTION.            RED  332
C                 SET TO ANY OTHER INTEGER IF BANDWIDTH REDUCTION IS     RED  333
C                 MORE IMPORTANT THAN PROFILE REDUCTION.                 RED  334
C  EXPLANATION OF OUTPUT VARIABLES--                                      RED  340
                                  .
                                  .
                                  .
      INTEGER STNODE, RVNODE, RENUM, XC, SORT2, STNUM, CCSTOR,           RED  560
     * SIZE, STPT, SBNUM                                                 RED  570
      INTEGER OPTPRO                                                     RED  571
      COMMON /GRA/ N, IDPTH, IDEG                                        RED  580
                                  .
                                  .
                                  .
      CALL FNDIAM(STNODE, RVNODE, NDSTK, NR, NDEG, LVL, LVLS1,           RED  960
     * LVLS2, CCSTOR, IDFLT)                                             RED  970
      IF (OPTPRO.EQ.1) GO TO 50                                          RED  971
      IF (NDEG(STNODE).LE.NDEG(RVNODE)) GO TO 50                         RED  980
                                  .
                                  .
                                  .
   70 ISDIR = ISDIR*NFLG                                                 RED 1220
      NUM = SBNUM                                                        RED 1230
C IF PROFILE REDUCTION IS MORE IMPORTANT CALL PROFIT INSTEAD OF          RED 1231
C NUMBER AND IGNORE ISDIR.                                               RED 1232
      IF (OPTPRO.EQ.1) GO TO 90                                          RED 1233
      IF (ISDIR.LT.Ø) NUM = STNUM                                        RED 1240
                                  .
                                  .
                                  .
C IF ORIGINAL NUMBERING IS BETTER THAN NEW ONE, SET UP TO RETURN IT      RED 1320
   75 DO 80 I=1,N                                                        RED 1330
        RENUM(I) = IOLD(I)                                               RED 1340
                                  .
                                  .
                                  .
      RETURN                                                             RED 1380
```

```
   90 CALL PROFIT(NR, NDSTK, RENUM, NDEG, LVLS2, LVLS1, LVL, NUM)      RED 1381
      SBNUM = NUM                                                      RED 1382
      IF (SBNUM.LE.STNUM) GO TO 30                                     RED 1383
C SOMETIMES PROFILE REDUCTION IS IMPROVED BY REVERSING THE NUMBERING   RED 1384
C PRODUCED BY PROFIT.  SUBROUTINE CHECK DETERMINES IF THIS IS THE CASE. RED 1385
      CALL CHECK(IBW2, IPF2, RENUM, NDSTK, NR, NDEG, LVL)              RED 1386
C IF ORIGINAL NUMBERING GIVES BETTER PROFILE USE IT RATHER THAN RENUM. RED 1387
      IF (IPF2.LE.IPF1) RETURN                                         RED 1388
      GO TO 75                                                         RED 1389
      END'                                                             RED 1390
```

ACM Transactions on Mathematical Software, Vol. 8, No. 2, June 1982, Page 221.

REMARK ON ALGORITHMS 508 AND 509

Matrix Bandwidth and Profile Reduction [H.L. Crane, Jr., N.E. Gibbs, W.G. Poole, Jr., and P.K. Stockmeyer, *ACM Trans. Math. Softw. 2*, 4 (Dec. 1976), 375–377] and A Hybrid Profile Reduction Algorithm [N.E. Gibbs, *ACM Trans. Math. Softw. 2*, 4 (Dec. 1976), 378–387]

John G. Lewis [Received 28 May 1980; revised 6 January 1982; accepted 10 January 1982]

Boeing Computer Services Co., Mail Stop 9C-01, 565 Andover Park West, Tukwila, WA 98188.

A new implementation of the Gibbs–Poole–Stockmeyer and Gibbs–King algorithms is available as Algorithm 582. This new implementation is faster, more robust, and requires less storage than the previous implementation of the Gibbs–Poole–Stockmeyer algorithm [1], distributed as Algorithm 508, and of the Gibbs–King algorithm [2], distributed as Algorithm 509. The mathematical capabilities of Algorithm 582 are identical to those of Algorithms 508 and 509. References [3] and [4] give the implementation details and documentation for the new implementation.

REFERENCES

1. CRANE, H.L., JR., GIBBS, N.E., POOLE, W. G., JR., AND STOCKMEYER, P.K. Matrix bandwidth and profile reduction. *ACM Trans. Math. Softw. 2*, 4 (Dec. 1976), 375–377.
2. GIBBS, N.E. A hybrid profile reduction algorithm. *ACM Trans. Math. Softw. 2*, 4 (Dec. 1976), 378–387.
3. LEWIS, J.G. Implementation of the Gibbs–Poole–Stockmeyer and Gibbs–King algorithms. *ACM Trans. Math. Softw. 8*, 2 (June 1982), 180–189.
4. LEWIS, J.G. Algorithm 582. The Gibbs–Poole–Stockmeyer and Gibbs–King algorithms for reordering Sparse matrices. *ACM Trans. Math. Softw. 8*, 2 (June 1982), 190–194.

Algorithm 510
Piecewise Linear Approximations to Tabulated Data [E2]

D. G. WILSON
Oak Ridge National Laboratory

Key Words and Phrases: approximation, first degree spline, linear, piecewise linear
CR Categories: 5.13
Language: Fortran

DESCRIPTION

This algorithm generates a piecewise linear approximation to tabulated data which is within specified tolerances of the data points. There are eight options available. The approximation may be required to be continuous or not, a single tolerance or a tolerance for each data point may be specified, and the approximation may or may not be restricted to the piecewise linear "tolerance band" centered at the data points with edges determined by the tolerances at the data points. Among approximations of the kind requested, the algorithm gives one of fewest line segments.

The algorithm processes data in order of increasing abscissa values, successively finding the longest approximating segment by a systematic search technique. This algorithm is a descendent of the one given in [1] and the systematic search technique is the same. The specified options determine where a new segment begins and whether or not successive segments intersect.

Even if a continuous approximation is not requested, the algorithm determines whether or not successive segments would intersect at an acceptable point if the partition point between them were adjusted. If this is the case, the adjustment is made. This means that the algorithm may give a continuous approximation when one was not requested even in cases where the algorithm of [1] would not. However it does not mean that the algorithm will necessarily give a continuous approximation when one is not requested even though there may exist a continuous approximation of the same number of line segments.

In many cases there is considerable freedom of choice in determining the last segment or two of the approximation. The criteria used to make these choices are continuity first and nearness of approach to the data second.

The algorithm will accept any nonnegative tolerances. If all tolerances are zero, then the approximation linearly interpolates the data.

When the approximation is required to be continuous and not restricted to the tolerance band, the algorithm gives a first degree spline which approximates the data within the specified tolerances and is of fewest line segments.

Received 14 December 1972.

Author's address: Union Carbide Corp., Nuclear Division, P.O. Box Y, Oak Ridge, TN 37830.
[The author is in the Computer Sciences Division at Oak Ridge Laboratory, which is operated by Union Carbide Corp.]

The inputs and outputs including the form of the approximation generated are described after the calling sequence, which is:

CALL STL2(X,Y,E,M,U,V,W,K,IP)

X, Y, E, and M contain input data; U, V, K, and possibly W contain output; and IP contains a parameter which specifies the options desired.

X and Y are real arrays of M elements. X(I) and Y(I) contain the abscissa and ordinate respectively of the I-th data point. M is an integer variable containing the dimension of the arrays X and Y.

E may be a single real variable or an M-element array depending on the value of IP. E contains the nonnegative tolerance or tolerances. If E is an array, then E(I) is the tolerance associated with (X(I), Y(I)) for I = 1, . . . , M.

U and V are real arrays of at least K + 1 elements. (Since K, the number of approximating segments, is not known a priori, the dimension of U and V must be "sufficiently large.") After STL2 has executed, U will contain a partition of the interval [X(1),X(N)] with U(1) = X(1) and U(K + 1) = X(N). V(I) is an ordinate to be associated with U(I) in the approximation. If a continuous approximation is requested, then V(I) is "the" ordinate to be associated with U(I).

If a continuous approximation is requested, then W is not used. In this case the I-th approximating segment is the straight line from (U(I),V(I)) to (U(I + 1), V(I + 1)).

If a continuous approximation is not requested, then W is a real array of at least K elements. In this case the I-th approximating segment is the straight line from (U(I),W(I)) to (U(I + 1),V(I + 1)), and V(1) is set equal to W(1).

K is an integer variable. After STL2 has executed, K will contain either the number of approximating line segments or zero. K will be set to zero in case of an error return.

Call STL2(X,Y,E,M,X,Y,E,M,IP) will not cause problems provided that either a continuous approximation is requested, or E is a sufficiently large real array.

IP is an integer variable. The content of IP is the product of three indicators: I1, I2, and I3. I1 indicates whether or not E is an array of tolerances. I1 = −1 indicates that E is an array. I1 = +1 indicates that E is a single variable. I2 indicates whether or not the approximation is to be restricted to the tolerance band about the data. I2 = 1 indicates that the approximation is *not* to be restricted to the tolerance band. I2 = 2 indicates that the restriction *is* to be so restricted. I3 indicates whether or not the approximation is required to be continuous. I3 = 1 indicates that the approximation *need not be* continuous. I3 = 3 indicates that the approximation *must be* continuous.

The program performs the following data checks: Are the abscissa values stored in order of increasing magnitude? Is the tolerance (or are the tolerances) nonnegative? Is the number of data points greater than 1? If the answer to any of these questions is no, then the program returns with K set to zero. In this case no further processing is attempted.

TEST RESULTS

This algorithm has successfully approximated 10 sets of data specifically designed to test various parts of the program. The main routine was written to repeat each data set with the ordinate values replaced by their negatives. This provides a new approximation problem for the algorithm, essentially doubling the number of data sets, but one for which the result is known. In addition, each data set was run with each possible option. Thus roughly 150 approximations have been generated for data sets of typically 12 points.

These tests were performed on IBM 360 model 75 and model 91 computers. Execution time is highly dependent on the data; however the model 91 processed 80 approximation problems of 10 to 12 points each in 2.04 seconds. Since the processing associated with terminating and initiating segments is nonnegligible and since in these constructed test cases the ratio of approximating segments to data points was almost .5, one would expect the program to do slightly better for smoother data.

REFERENCES

1. WILSON, D.G. Piecewise linear approximations of fewest line segments. Proc. AFIPS 1972 SJCC, Vol. 40, AFIPS Press, Montvale, N.J., pp. 187-198.

ALGORITHM

```
      SUBROUTINE STL2(X, Y, E, M, U, V, W, K, IP)                    STL  1Ø
C PIECEWISE LINEAR APPROXIMATIONS OF FEWEST                          STL  2Ø
C LINE SEGMENTS WITHIN GIVEN TOLERANCES.                             STL  3Ø
C X,Y,E AND M CONTAIN INPUT DATA.                                    STL  4Ø
C U,V,K AND POSSIBLY W CONTAIN OUTPUT.                               STL  5Ø
C IP IS A PARAMETER DETERMINING THE OPERATION                        STL  6Ø
C  OF THE PROGRAM.                                                  STL  7Ø
C X AND Y ARE INPUT DATA ARRAYS OF M ELEMENTS                        STL  8Ø
C  X(I),Y(I) CONTAINS THE ITH DATA POINT.                           STL  9Ø
C E MAY BE A SINGLE TOLERANCE OR A TABLE OF                          STL  1ØØ
C  TOLERANCES DEPENDING ON THE VALUE OF IP.                          STL  11Ø
C IF E IS AN ARRAY, THEN E(I) IS THE TOLERANCE                       STL  12Ø
C  ASSOCIATED WITH X(I),Y(I) AND E MUST CONTAIN                      STL  13Ø
C  M NONNEGATIVE ELEMENTS.                                           STL  14Ø
C U AND V ARE OUTPUT ARRAYS OF K+1 ELEMENTS.                         STL  15Ø
C  U IS A PARTITION OF THE INTERVAL (X(1),X(N))                      STL  16Ø
C  WITH U(1)=X(1) AND U(K+1)=X(N).                                   STL  17Ø
C  V(I) IS AN ORDINATE TO BE ASSOCIATED WITH                         STL  18Ø
C  U(I) IN THE APPROXIMATION.  (IF A CONTINUOUS                      STL  19Ø
C  APPROXIMATION IS REQUESTED, THEN V(I) IS                          STL  2ØØ
C  'THE' ORDINATE TO BE ASSOCIATED WITH U(I).)                       STL  21Ø
C IF A CONTINUOUS APPROXIMATION IS REQUESTED,                        STL  22Ø
C  THEN W IS NOT USED.  IN THIS CASE THE ITH                         STL  23Ø
C  APPROXIMATING SEGMENT IS THE STRAIGHT LINE                        STL  24Ø
C  FROM U(I),V(I) TO U(I+1),V(I+1).                                  STL  25Ø
C IF A CONTINUOUS APPROXIMATION IS NOT                               STL  26Ø
C  REQUESTED, THEN W IS A K-ELEMENT OUTPUT                           STL  27Ø
C  ARRAY.  IN THIS CASE THE ITH APPROXIMATING                        STL  28Ø
C  SEGMENT IS THE STRAIGHT LINE FROM                                 STL  29Ø
C  U(I),W(I) TO U(I+1),V(I+1), AND V(1) IS                           STL  3ØØ
C  SET EQUAL TO W(1).                                                STL  31Ø
C K IS THE NUMBER OF SEGMENTS IN THE PIECE-                          STL  32Ø
C  WISE LINEAR APPROXIMATION GENERATED.  IN                          STL  33Ø
C  CASE OF AN ERROR RETURN, K WILL BE SET TO                         STL  34Ø
C  ZERO.                                                             STL  35Ø
C THE CONTROL PARAMETER IP IS THE PRODUCT                            STL  36Ø
C  OF THREE INDICATORS I1,I2 AND I3.                                 STL  37Ø
C  I1 INDICATES WHETHER OR NOT E IS AN                               STL  38Ø
C     ARRAY OF TOLERANCES.                                           STL  39Ø
C     I1 = -1 INDICATES E IS AN ARRAY                                STL  4ØØ
C     I1 = +1 INDICATES E IS A SINGLE NUMBER.                        STL  41Ø
C  I2 INDICATES WHETHER OR NOT THE                                   STL  42Ø
C     APPROXIMATION IS TO BE RESTRICTED TO                           STL  43Ø
C     THE 'TOLERANCE BAND' ABOUT THE DATA.                           STL  44Ø
C     I2 = 1 INDICATES NO BAND RESTRICTION                           STL  45Ø
C     I2 = 2 INDICATES APPLY THIS RESTRICTION                        STL  46Ø
C     (THE 'TOLERANCE BAND' IS A PIECEWISE                           STL  47Ø
C     LINEAR BAND CENTERED AT THE DATA WHOSE                         STL  48Ø
C     WIDTH IS DETERMINED BY THE TOLERANCES                          STL  49Ø
C     AT THE DATA POINTS.)                                           STL  5ØØ
C  I3 INDICATES WHETHER OR NOT THE                                   STL  51Ø
C     APPROXIMATION MUST BE CONTINUOUS.                              STL  52Ø
C     I3 = 1 INDICATES CONTINUITY NOT REQUIRED                       STL  53Ø
C     I3 = 3 INDICATES CONTINUITY IS REQUIRED                        STL  54Ø
C CALL STL2 (X,Y,E,M,X,Y,E,M,IP) WILL NOT                            STL  55Ø
C  CAUSE PROBLEMS PROVIDED THAT                                      STL  56Ø
C  EITHER A CONTINUOUS APPROXIMATION IS                              STL  57Ø
C  REQUESTED, OR E IS A SUFFICIENTLY LARGE                           STL  58Ø
C  ARRAY.                                                            STL  59Ø
C THE PROGRAM PERFORMS THE FOLLOWING DATA                            STL  6ØØ
C  CHECKS.  ARE THE X-VALUES IN INCREASING                           STL  61Ø
C  ORDER.  ARE THE TOLERANCE(S) NONNEGATIVE.                         STL  62Ø
C  IS THE NUMBER OF DATA POINTS GREATER THAN                         STL  63Ø
C  ONE.  IF ANY CHECK FAILS, THE PROGRAM                             STL  64Ø
C  RETURNS WITH K SET EQUAL TO Ø.  IN THIS                           STL  65Ø
C  CASE NO FURTHER PROCESSING IS ATTEMPTED.                          STL  66Ø
      DIMENSION X(9), Y(9), E(9), U(9), V(9), W(9)                   STL  67Ø
      N = M                                                          STL  68Ø
      ITCH = IP                                                      STL  69Ø
      J = 1                                                          STL  7ØØ
```

```
C ERROR CHECKS                                                          STL  71Ø
      IF (N.LE.1) GO TO 4ØØ                                             STL  72Ø
      IF (E(1).LT.Ø.Ø) GO TO 4ØØ                                        STL  73Ø
      DO 1Ø L=2,N                                                       STL  74Ø
        IF (X(L-1).GE.X(L)) GO TO 4ØØ                                   STL  75Ø
        IF (ITCH.GE.Ø) GO TO 1Ø                                         STL  76Ø
        IF (E(L).LT.Ø.Ø) GO TO 4ØØ                                      STL  77Ø
   1Ø CONTINUE                                                          STL  78Ø
C INITIALIZATION FOR ENTIRE PROGRAM                                     STL  79Ø
      EPSLN = E(1)                                                      STL  8ØØ
      SGN = 1.Ø                                                         STL  81Ø
      KEEP = 1                                                          STL  82Ø
      I = 1                                                             STL  83Ø
      U(1) = X(1)                                                       STL  84Ø
      J = 2                                                             STL  85Ø
      INIT = 1                                                          STL  86Ø
      INDC = Ø                                                          STL  87Ø
      GO TO 3Ø                                                          STL  88Ø
C INITIALIZATION FOR EACH SEGMENT                                       STL  89Ø
   2Ø CONTINUE                                                          STL  9ØØ
      J = J + 1                                                         STL  91Ø
      INIT = I                                                          STL  92Ø
      INDC = Ø                                                          STL  93Ø
      IF (IABS(ITCH).LT.3) KEEP = I                                     STL  94Ø
      IF (IABS(IABS(ITCH)-4).NE.2) GO TO 3Ø                             STL  95Ø
C RESTRICTED TO TOLERANCE BAND                                          STL  96Ø
      XEYE = U(J-1)                                                     STL  97Ø
      YEYE = V(J-1)                                                     STL  98Ø
      TEMP1 = EPSLN                                                     STL  99Ø
      IF (ITCH.LT.Ø) TEMP1 = TEMP1 + (SGN*E(I-1)-EPSLN)*(X(I)-U(J-1)    STL 1ØØØ
     * )/(X(I)-X(I-1))                                                 STL 1Ø1Ø
      YINIT = YEYE - TEMP1 - TEMP1                                      STL 1Ø2Ø
      GO TO 4Ø                                                          STL 1Ø3Ø
   3Ø CONTINUE                                                          STL 1Ø4Ø
C NOT RESTRICTED TO TOLERANCE BAND                                      STL 1Ø5Ø
      XEYE = X(I)                                                       STL 1Ø6Ø
      YEYE = Y(I) + EPSLN                                               STL 1Ø7Ø
      YINIT = Y(I) - EPSLN                                              STL 1Ø8Ø
      IF (IABS(ITCH).EQ.1 .OR. I.EQ.1) GO TO 4Ø                         STL 1Ø9Ø
      TEMP1 = EPSLN                                                     STL 11ØØ
      IF (ITCH.LT.Ø) TEMP1 = SGN*E(I+1)                                 STL 111Ø
      SMIN = (Y(I+1)-YEYE-TEMP1)/(X(I+1)-XEYE)                          STL 112Ø
      IF (ITCH.LT.Ø) TEMP1 = SGN*E(I-1)                                 STL 113Ø
      SMAX = (YEYE-Y(I-1)+TEMP1)/(XEYE-X(I-1))                          STL 114Ø
      IF (KEEP.EQ.I-1) GO TO 5Ø                                         STL 115Ø
      IT = I - 2                                                        STL 116Ø
      XINIT = XEYE                                                      STL 117Ø
      IPIV = I                                                          STL 118Ø
      IGRAZE = I                                                        STL 119Ø
      I = I + 1                                                         STL 12ØØ
      GO TO 15Ø                                                         STL 121Ø
   4Ø CONTINUE                                                          STL 122Ø
      IF (XEYE.GE.X(I)) I = I + 1                                       STL 123Ø
      IF (ITCH.LT.Ø) EPSLN = SGN*E(I)                                   STL 124Ø
      DX = X(I) - XEYE                                                  STL 125Ø
      SMAX = (Y(I)+EPSLN-YEYE)/DX                                       STL 126Ø
      SMIN = (Y(I)-EPSLN-YEYE)/DX                                       STL 127Ø
   5Ø CONTINUE                                                          STL 128Ø
      XINIT = XEYE                                                      STL 129Ø
      IPIV = I                                                          STL 13ØØ
      IGRAZE = I                                                        STL 131Ø
C DETERMINATION OF INDIVIDUAL SEGMENT                                   STL 132Ø
   6Ø CONTINUE                                                          STL 133Ø
      IF (I.EQ.N) GO TO 26Ø                                             STL 134Ø
      I = I + 1                                                         STL 135Ø
   7Ø CONTINUE                                                          STL 136Ø
C TEST FOR NEW *MAX* SLOPE                                              STL 137Ø
      DX = X(I) - XEYE                                                  STL 138Ø
      IF (ITCH.LT.Ø) EPSLN = SGN*E(I)                                   STL 139Ø
      TEMP1 = (Y(I)+EPSLN-YEYE)/DX                                      STL 14ØØ
      TEST = TEMP1 - SMAX                                               STL 141Ø
      IF (SGN.LE.Ø.Ø) TEST = -TEST                                      STL 142Ø
      IF (TEST) 8Ø, 9Ø, 1ØØ                                             STL 143Ø
   8Ø CONTINUE                                                          STL 144Ø
C TEST FOR END OF CANDIDATE SEGMENT                                     STL 145Ø
```

```
      TEST = TEMP1 - SMIN                                          STL 146Ø
      IF (SGN.LE.Ø.Ø) TEST = -TEST                                 STL 147Ø
      IF (TEST.LT.Ø.Ø) GO TO 21Ø                                   STL 148Ø
      SMAX = TEMP1                                                 STL 149Ø
   9Ø CONTINUE                                                     STL 15ØØ
C TEST FOR NEW *MIN* SLOPE                                         STL 151Ø
      IPIV = I                                                     STL 152Ø
  1ØØ CONTINUE                                                     STL 153Ø
      TEMP2 = (Y(I)-EPSLN-YEYE)/DX                                 STL 154Ø
      TEST = TEMP2 - SMAX                                          STL 155Ø
      IF (SGN.LE.Ø.Ø) TEST = -TEST                                 STL 156Ø
      IF (TEST) 11Ø, 12Ø, 14Ø                                      STL 157Ø
  11Ø CONTINUE                                                     STL 158Ø
      TEST = SMIN - TEMP2                                          STL 159Ø
      IF (SGN.LE.Ø.Ø) TEST = -TEST                                 STL 16ØØ
      IF (TEST) 12Ø, 13Ø, 6Ø                                       STL 161Ø
  12Ø CONTINUE                                                     STL 162Ø
      SMIN = TEMP2                                                 STL 163Ø
  13Ø CONTINUE                                                     STL 164Ø
      IGRAZE = I                                                   STL 165Ø
      GO TO 6Ø                                                     STL 166Ø
C CHECK FOR PIVOT AT NEW EYE POINT                                 STL 167Ø
  14Ø CONTINUE                                                     STL 168Ø
      IF (XEYE.EQ.X(IPIV)) GO TO 22Ø                               STL 169Ø
      IF (ITCH.LT.Ø) EPSLN = SGN*E(IPIV)                           STL 17ØØ
      INDC = 1                                                     STL 171Ø
      SVX = XEYE                                                   STL 172Ø
      SVY = YEYE                                                   STL 173Ø
      SVMN = SMIN                                                  STL 174Ø
      SVMX = SMAX                                                  STL 175Ø
      XEYE = X(IPIV)                                               STL 176Ø
      YEYE = Y(IPIV) + EPSLN                                       STL 177Ø
      SMIN = SMAX                                                  STL 178Ø
      SMAX = (YINIT-YEYE)/(XINIT-XEYE)                             STL 179Ø
      IF (KEEP.GE.IPIV) GO TO 17Ø                                  STL 18ØØ
      IT = IPIV - 1                                                STL 181Ø
  15Ø CONTINUE                                                     STL 182Ø
      TEMP2 = YEYE + EPSLN                                         STL 183Ø
      DO 16Ø L=KEEP,IT                                             STL 184Ø
        IF (ITCH.LT.Ø) TEMP2 = YEYE + SGN*E(L)                     STL 185Ø
        TEMP1 = (Y(L)-TEMP2)/(X(L)-XEYE)                           STL 186Ø
        TEST = TEMP1 - SMAX                                        STL 187Ø
        IF (SGN.LE.Ø.Ø) TEST = -TEST                               STL 188Ø
        IF (TEST.LT.Ø.Ø) SMAX = TEMP1                              STL 189Ø
  16Ø CONTINUE                                                     STL 19ØØ
  17Ø CONTINUE                                                     STL 191Ø
      IF (IPIV.GE.I-1) GO TO 7Ø                                    STL 192Ø
      IT = I - 2                                                   STL 193Ø
      TEMP2 = YEYE - EPSLN                                         STL 194Ø
      IDIOT = IPIV                                                 STL 195Ø
      DO 2ØØ L=IDIOT,IT                                            STL 196Ø
        DX = X(L+1) - XEYE                                         STL 197Ø
        IF (ITCH.LT.Ø) TEMP2 = YEYE - SGN*E(L+1)                   STL 198Ø
        TEMP1 = (Y(L+1)-TEMP2)/DX                                  STL 199Ø
        TEST = TEMP1 - SMAX                                        STL 2ØØØ
        IF (SGN.LE.Ø.Ø) TEST = -TEST                               STL 2Ø1Ø
        IF (TEST) 18Ø, 19Ø, 2ØØ                                    STL 2Ø2Ø
  18Ø   CONTINUE                                                   STL 2Ø3Ø
        SMAX = TEMP1                                               STL 2Ø4Ø
  19Ø   CONTINUE                                                   STL 2Ø5Ø
        IPIV = L + 1                                               STL 2Ø6Ø
  2ØØ CONTINUE                                                     STL 2Ø7Ø
      GO TO 7Ø                                                     STL 2Ø8Ø
C END OF CURRENT SEGMENT                                           STL 2Ø9Ø
  21Ø CONTINUE                                                     STL 21ØØ
      TEMP2 = SMIN                                                 STL 211Ø
      IF (I.EQ.N) GO TO 24Ø                                        STL 212Ø
      KEEP = IGRAZE                                                STL 213Ø
      GO TO 25Ø                                                    STL 214Ø
  22Ø CONTINUE                                                     STL 215Ø
      TEMP2 = SMAX                                                 STL 216Ø
      IF (I.EQ.N) GO TO 23Ø                                        STL 217Ø
      SGN = -SGN                                                   STL 218Ø
      EPSLN = -EPSLN                                               STL 219Ø
      KEEP = IPIV                                                  STL 22ØØ
      GO TO 25Ø                                                    STL 221Ø
```

```
  23Ø CONTINUE                                                                  STL 222Ø
      IF (INDC.EQ.Ø .OR. XEYE.NE.X(N-1)) GO TO 24Ø                              STL 223Ø
      XEYE = SVX                                                                STL 224Ø
      YEYE = SVY                                                                STL 225Ø
      SMIN = SVMN                                                               STL 226Ø
      SMAX = SVMX                                                               STL 227Ø
  24Ø CONTINUE                                                                  STL 228Ø
      U(J) = X(N-1)                                                             STL 229Ø
      YINIT = Y(N-1)                                                            STL 23ØØ
      GO TO 27Ø                                                                 STL 231Ø
  25Ø CONTINUE                                                                  STL 232Ø
      IF (IABS(IABS(ITCH)-4).NE.2) GO TO 3ØØ                                    STL 233Ø
C DETERMINE KNOT ON EDGE OF TOLERANCE BAND                                      STL 234Ø
      TEMP1 = Ø.Ø                                                               STL 235Ø
      IF (ITCH.LT.Ø) TEMP1 = EPSLN - SGN*E(I-1)                                 STL 236Ø
      TEMP1 = (Y(I)-Y(I-1)+TEMP1)/(X(I)-X(I-1))                                 STL 237Ø
      U(J) = (Y(I)+EPSLN-YEYE-TEMP1*X(I)+TEMP2*XEYE)/(TEMP2-TEMP1)              STL 238Ø
      GO TO 31Ø                                                                 STL 239Ø
  26Ø CONTINUE                                                                  STL 24ØØ
      U(J) = X(N)                                                               STL 241Ø
      YINIT = Y(N)                                                              STL 242Ø
  27Ø CONTINUE                                                                  STL 243Ø
C CONTINUITY CHECK FOR LAST SEGMENT                                             STL 244Ø
      IF (IABS(ITCH).GE.3 .OR. INIT.EQ.1) GO TO 29Ø                             STL 245Ø
      IT = INIT - 1                                                             STL 246Ø
      SVMX = SMAX + SGN                                                         STL 247Ø
      TEMP2 = YEYE + EPSLN                                                      STL 248Ø
      DO 28Ø L=KP,IT                                                            STL 249Ø
        IF (ITCH.LT.Ø) TEMP2 = YEYE + SGN*E(L)                                  STL 25ØØ
        TEMP1 = (Y(L)-TEMP2)/(X(L)-XEYE)                                        STL 251Ø
        TEST = TEMP1 - SVMX                                                     STL 252Ø
        IF (SGN.LE.Ø.Ø) TEST = -TEST                                            STL 253Ø
        IF (TEST.LT.Ø.Ø) SVMX = TEMP1                                           STL 254Ø
  28Ø CONTINUE                                                                  STL 255Ø
      IF (ABS(SVMX-SMAX+SVMX-SMIN).LE.ABS(SMAX-SMIN)) SMAX = SVMX               STL 256Ø
  29Ø CONTINUE                                                                  STL 257Ø
C NEARNESS CHECK FOR LAST SEGMENT                                               STL 258Ø
      TEMP2 = SMAX                                                              STL 259Ø
      TEMP1 = YEYE + SMAX*(U(J)-XEYE)                                           STL 26ØØ
      TEST = YINIT - TEMP1                                                      STL 261Ø
      IF (SGN.LT.Ø.Ø) TEST = -TEST                                              STL 262Ø
      IF (TEST.GT.Ø.Ø) GO TO 31Ø                                                STL 263Ø
      TEMP2 = SMIN                                                              STL 264Ø
      TEMP1 = YEYE + SMIN*(U(J)-XEYE)                                           STL 265Ø
      TEST = YINIT - TEMP1                                                      STL 266Ø
      IF (SGN.LT.Ø.Ø) TEST = -TEST                                              STL 267Ø
      IF (TEST.LT.Ø.Ø) GO TO 31Ø                                                STL 268Ø
      TEMP2 = (YINIT-YEYE)/(U(J)-XEYE)                                          STL 269Ø
      V(J) = YINIT                                                              STL 27ØØ
      GO TO 32Ø                                                                 STL 271Ø
  3ØØ CONTINUE                                                                  STL 272Ø
      IF (IABS(ITCH).GE.3) GO TO 33Ø                                            STL 273Ø
      U(J) = Ø.5*(X(I)+X(I-1))                                                  STL 274Ø
  31Ø CONTINUE                                                                  STL 275Ø
      V(J) = YEYE + TEMP2*(U(J)-XEYE)                                           STL 276Ø
  32Ø CONTINUE                                                                  STL 277Ø
      IF (XEYE.NE.XINIT) GO TO 33Ø                                              STL 278Ø
      IF (IABS(ITCH).EQ.2) GO TO 36Ø                                            STL 279Ø
      IF (IABS(ITCH).NE.6) GO TO 33Ø                                            STL 28ØØ
      IF (J.LE.2) GO TO 38Ø                                                     STL 281Ø
      GO TO 39Ø                                                                 STL 282Ø
  33Ø CONTINUE                                                                  STL 283Ø
C RECOMPUTATION OF KNOT FOR CONTINUITY                                          STL 284Ø
      IF (J.LE.2) GO TO 37Ø                                                     STL 285Ø
      IF (SLOPE.EQ.TEMP2) GO TO 36Ø                                             STL 286Ø
      YINIT = V(J-2)                                                            STL 287Ø
      IF (IABS(ITCH).LT.3) YINIT = W(J-2)                                       STL 288Ø
      TEMP1 = (XEYE*TEMP2-U(J-2)*SLOPE+YINIT-YEYE)/(TEMP2-SLOPE)                STL 289Ø
      IF (IABS(ITCH).GE.3) GO TO 35Ø                                            STL 29ØØ
      IF (TEMP1.GT.XINIT) GO TO 36Ø                                             STL 291Ø
      TEST = ABS(EPSLN)                                                         STL 292Ø
      IDIOT = INIT - KP                                                         STL 293Ø
      DO 34Ø L=1,IDIOT                                                          STL 294Ø
        IT = INIT - L                                                           STL 295Ø
        IF (TEMP1.GE.X(IT)) GO TO 35Ø                                           STL 296Ø
        DX = Y(IT) - YEYE - TEMP2*(X(IT)-XEYE)                                  STL 297Ø
```

```
      IF (ITCH.LT.Ø) TEST = E(IT)                                    STL 298Ø
      IF (ABS(DX).GT.TEST) GO TO 36Ø                                 STL 299Ø
34Ø CONTINUE                                                         STL 3ØØØ
35Ø CONTINUE                                                         STL 3Ø1Ø
    U(J-1) = TEMP1                                                   STL 3Ø2Ø
    V(J-1) = YEYE + TEMP2*(U(J-1)-XEYE)                              STL 3Ø3Ø
    IF (IABS(ITCH).LT.3) W(J-1) = V(J-1)                             STL 3Ø4Ø
    GO TO 39Ø                                                        STL 3Ø5Ø
36Ø CONTINUE                                                         STL 3Ø6Ø
    W(J-1) = YEYE + TEMP2*(U(J-1)-XEYE)                              STL 3Ø7Ø
    GO TO 39Ø                                                        STL 3Ø8Ø
37Ø CONTINUE                                                         STL 3Ø9Ø
    IF (IABS(ITCH).LT.3) GO TO 36Ø                                   STL 31ØØ
38Ø CONTINUE                                                         STL 311Ø
    V(1) = YEYE + TEMP2*(U(1)-XEYE)                                  STL 312Ø
39Ø CONTINUE                                                         STL 313Ø
    SLOPE = TEMP2                                                    STL 314Ø
    KP = KEEP                                                        STL 315Ø
    IF (I.LT.N) GO TO 2Ø                                             STL 316Ø
    IF (X(N).EQ.U(J)) GO TO 4ØØ                                      STL 317Ø
    IF (IABS(ITCH).LT.3) W(J) = V(J)                                 STL 318Ø
    J = J + 1                                                        STL 319Ø
    U(J) = X(N)                                                      STL 32ØØ
    V(J) = Y(N)                                                      STL 321Ø
4ØØ CONTINUE                                                         STL 322Ø
    IF (J.GE.2 .AND. IABS(ITCH).LT.3) V(1) = W(1)                    STL 323Ø
    K = J - 1                                                        STL 324Ø
    RETURN                                                           STL 325Ø
    END                                                              STL 326Ø
```

ALGORITHM 511
CDC 6600 Subroutines IBESS and JBESS for Bessel Functions $I_\nu(x)$ and $J_\nu(x)$, $x \geq 0$, $\nu \geq 0$ [S18]

D. E. AMOS, S. L. DANIEL, and M. K. WESTON
Sandia Laboratories

Key Words and Phrases: Bessel functions of the first kind, modified Bessel function, Airy functions, uniform asymptotic expansion
CR Categories: 5.12
Language: Fortran

DESCRIPTION

This algorithm is a complement to [1], where the theoretical background and development are described.

REFERENCES

1. AMOS, D.E., DANIEL, S.L., AND WESTON, M.K. CDC 6600 Subroutines IBESS and JBESS for Bessel Functions $I_\nu(x)$ and $J_\nu(x)$, $x \geq 0, \nu \geq 0$. *ACM Trans. Math. Software 3*, 1 (March 1977), 76-92.

ALGORITHM

```
      FUNCTION GAMLN(X)                                                  GAM   1Ø
C                                                                        GAM   2Ø
C         A CDC 66ØØ SUBROUTINE                                          GAM   3Ø
C                                                                        GAM   4Ø
C     AUTHORS                                                            GAM   5Ø
C         D.E. AMOS AND S.L. DANIEL                                      GAM   6Ø
C         ALBUQUERQUE, NEW MEXICO, 87115                                 GAM   7Ø
C         JANUARY, 1975                                                  GAM   8Ø
C                                                                        GAM   9Ø
C     REFERENCES                                                         GAM  1ØØ
C         ABRAMOWITZ, M. AND STEGUN, I.A. HANDBOOK OF MATHEMATICAL       GAM  11Ø
C         FUNCTIONS. NBS APPLIED MATHEMATICS SERIES 55, U.S. GOVERNMENT  GAM  12Ø
C         PRINTING OFFICE, WASHINGTON, D.C., CHAPTER 6.                  GAM  13Ø
C                                                                        GAM  14Ø
C         AMOS, D.E., DANIEL, S.L. AND WESTON, M.K. CDC 66ØØ             GAM  15Ø
C         SUBROUTINES IBESS AND JBESS FOR BESSEL FUNCTIONS               GAM  16Ø
C         I/SUB(NU)/(X) AND J/SUB(NU)/(X), X.GE.Ø, NU.GE.Ø.              GAM  17Ø
C         ACM TRANS. MATH. SOFTWARE, 1977.                               GAM  18Ø
C                                                                        GAM  19Ø
C         HART, J.F., ET. AL. COMPUTER APPROXIMATIONS, WILEY, NEW YORK.  GAM  2ØØ
C         PP. 13Ø-136, 1968.                                             GAM  21Ø
C                                                                        GAM  22Ø
C     ABSTRACT                                                           GAM  23Ø
```

Received 10 June 1974, 4 April 1975, and 6 January 1976.

Authors' address: Division 5122, Sandia Laboratories, Albuquerque, NM 87115.

```
C         GAMLN COMPUTES THE NATURAL LOG OF THE GAMMA FUNCTION FOR      GAM  24Ø
C         X.GT.Ø. A RATIONAL CHEBYSHEV APPROXIMATION IS USED ON         GAM  25Ø
C         8.LT.X.LT.1ØØØ., THE ASYMPTOTIC EXPANSION FOR X.GE.1ØØØ. AND  GAM  26Ø
C         BACKWARD RECURSION FOR Ø.LT.X.LT.8 FOR NON-INTEGRAL X. FOR    GAM  27Ø
C         X=1.,...,8., GAMLN IS SET TO NATURAL LOGS OF FACTORIALS.      GAM  28Ø
C                                                                       GAM  29Ø
C     DESCRIPTION OF ARGUMENTS                                          GAM  3ØØ
C                                                                       GAM  31Ø
C         INPUT                                                         GAM  32Ø
C           X      - X.GT.Ø                                           GAM  33Ø
C                                                                       GAM  34Ø
C         OUTPUT                                                        GAM  35Ø
C           GAMLN  - NATURAL LOG OF THE GAMMA FUNCTION AT X           GAM  36Ø
C                                                                       GAM  37Ø
C     ERROR CONDITIONS                                                  GAM  38Ø
C         IMPROPER INPUT ARGUMENT - A FATAL ERROR                       GAM  39Ø
      DOUBLE PRECISION DXZ, RXZ, XK                                     GAM  4ØØ
      DIMENSION G(8), P(5), Q(2)                                        GAM  41Ø
      DATA XLIM1,XLIM2,RTWPIL/    8.  ,  1ØØØ.   , 9.18938533204673E-Ø1/GAM  42Ø
      DATA  G               / 8.525161361Ø6541E+ØØ, 6.579251212Ø1Ø1ØE+ØØ,GAM  43Ø
     1 4.7874917427820Ø5E+ØØ, 3.178Ø5383Ø34795E+ØØ, 1.7917594692280Ø6E+ØØ,GAM  44Ø
     2 6.9314718Ø559945E-Ø1,         2(Ø.)       /                      GAM  45Ø
      DATA  P               / 7.663451880ØØØØØE-Ø4,-5.94Ø9561Ø52ØØØØE-Ø4,GAM  46Ø
     1 7.9364311Ø4845ØØE-Ø4,-2.77777775657725E-Ø3, 8.33333333333169E-Ø2/GAM  47Ø
      DATA  Q              /-2.77777777777778E-Ø3, 8.33333333333333E-Ø2/GAM  48Ø
      IF (X) 14Ø, 14Ø, 1Ø                                               GAM  49Ø
   1Ø DX = X - XLIM1                                                    GAM  5ØØ
      IF (DX) 2Ø, 11Ø, 8Ø                                               GAM  51Ø
   2Ø IF (X-1.) 3Ø, 13Ø, 4Ø                                             GAM  52Ø
   3Ø XZ = X + 8.                                                       GAM  53Ø
      TX = X                                                            GAM  54Ø
      FK = -.5                                                          GAM  55Ø
      NDX = 7                                                           GAM  56Ø
      GO TO 5Ø                                                          GAM  57Ø
   4Ø DX = ABS(DX)                                                      GAM  58Ø
      NDX = DX                                                          GAM  59Ø
      DNDX = NDX                                                        GAM  6ØØ
      NDX = NDX + 1                                                     GAM  61Ø
      IF ((DNDX-DX).EQ.Ø.) GO TO 12Ø                                    GAM  62Ø
      XZ = X + DNDX + 1.                                                GAM  63Ø
      TX = 1.                                                           GAM  64Ø
      FK = .5                                                           GAM  65Ø
   5Ø DXZ = XZ                                                          GAM  66Ø
      RXZ = 1.D+Ø/DXZ                                                   GAM  67Ø
      RX = RXZ                                                          GAM  68Ø
      RXX = RX*RX                                                       GAM  69Ø
      XK = 1.D+Ø                                                        GAM  7ØØ
      DO 6Ø I=1,NDX                                                     GAM  71Ø
        XK = XK - RXZ                                                   GAM  72Ø
        SXK = XK                                                        GAM  73Ø
        TX = TX*SXK                                                     GAM  74Ø
   6Ø CONTINUE                                                          GAM  75Ø
      SUM = (X-FK)*ALOG(XZ) - ALOG(TX) - XZ                             GAM  76Ø
      PX = P(1)                                                         GAM  77Ø
      DO 7Ø I=2,5                                                       GAM  78Ø
        PX = PX*RXX + P(I)                                              GAM  79Ø
   7Ø CONTINUE                                                          GAM  8ØØ
      GAMLN = PX*RX + SUM + RTWPIL                                      GAM  81Ø
      RETURN                                                            GAM  82Ø
   8Ø RX = 1./X                                                         GAM  83Ø
      RXX = RX*RX                                                       GAM  84Ø
      IF ((X-XLIM2).LT.Ø.) GO TO 9Ø                                     GAM  85Ø
      PX = Q(1)*RXX + Q(2)                                              GAM  86Ø
      GAMLN = PX*RX + (X-.5)*ALOG(X) - X + RTWPIL                       GAM  87Ø
      RETURN                                                            GAM  88Ø
   9Ø PX = P(1)                                                         GAM  89Ø
      SUM = (X-.5)*ALOG(X) - X                                          GAM  9ØØ
      DO 1ØØ I=2,5                                                      GAM  91Ø
        PX = PX*RXX + P(I)                                              GAM  92Ø
  1ØØ CONTINUE                                                          GAM  93Ø
      GAMLN = PX*RX + SUM + RTWPIL                                      GAM  94Ø
      RETURN                                                            GAM  95Ø
  11Ø GAMLN = G(1)                                                      GAM  96Ø
      RETURN                                                            GAM  97Ø
  12Ø GAMLN = G(NDX)                                                    GAM  98Ø
      RETURN                                                            GAM  99Ø
```

```
  13Ø GAMLN = G(8)                                                    GAM 1ØØØ
      RETURN                                                          GAM 1Ø1Ø
  14Ø PRINT 99999, X                                                  GAM 1Ø2Ø
      STOP                                                            GAM 1Ø3Ø
99999 FORMAT (5ØH ARGUMENT FOR GAMLN IS LESS THAN OR EQUAL TO ZERO,,  GAM 1Ø4Ø
     * 3H X=,E25.14)                                                  GAM 1Ø5Ø
      END                                                             GAM 1Ø6Ø


      SUBROUTINE IBESS(KODE, ALPHA, N, X, Y)                          IBE   1Ø
C                                                                     IBE   2Ø
C         A CDC 66ØØ SUBROUTINE                                       IBE   3Ø
C                                                                     IBE   4Ø
C     AUTHORS                                                         IBE   5Ø
C         D.E. AMOS AND S.L. DANIEL                                   IBE   6Ø
C         SANDIA LABORATORIES                                         IBE   7Ø
C         JANUARY, 1975                                               IBE   8Ø
C                                                                     IBE   9Ø
C     REFERENCES                                                      IBE  1ØØ
C         ABRAMOWITZ, M. AND STEGUN, I.A. HANDBOOK OF MATHEMATICAL    IBE  11Ø
C         FUNCTIONS. NBS APPLIED MATHEMATICS SERIES 55, U.S. GOVERNMENTIBE  12Ø
C         PRINTING OFFICE, WASHINGTON, D.C., CHAPTERS 9 AND 1Ø.       IBE  13Ø
C                                                                     IBE  14Ø
C         AMOS, D.E., DANIEL, S.L. AND WESTON, M.K. CDC 66ØØ          IBE  15Ø
C         SUBROUTINES IBESS AND JBESS FOR BESSEL FUNCTIONS            IBE  16Ø
C         I/SUB(NU)/(X) AND J/SUB(NU)/(X), X.GE.Ø, NU.GE.Ø.           IBE  17Ø
C         ACM TRANS. MATH. SOFTWARE, 1977.                            IBE  18Ø
C                                                                     IBE  19Ø
C         OLVER, F.W.J. TABLES FOR BESSEL FUNCTIONS OF MODERATE OR    IBE  2ØØ
C         LARGE ORDERS. NPL MATHEMATICAL TABLES, VOL 6. HER MAJESTY-S IBE  21Ø
C         STATIONERY OFFICE, LONDON, 1962.                            IBE  22Ø
C                                                                     IBE  23Ø
C         OLVER, F.W.J. THE ASYMPTOTIC EXPANSION OF BESSEL FUNCTIONS OFIBE  24Ø
C         LARGE ORDER. PHIL. TRANS. A, 247, PP. 328-368, 1954.        IBE  25Ø
C                                                                     IBE  26Ø
C     ABSTRACT                                                        IBE  27Ø
C         IBESS COMPUTES AN N MEMBER SEQUENCE OF I BESSEL FUNCTIONS   IBE  28Ø
C         I/SUB(ALPHA+K-1)/(X), K=1,...,N OR SCALED BESSEL FUNCTIONS  IBE  29Ø
C         EXP(-X)*I/SUB(ALPHA*K-1)/(X), K=1,...,N FOR NON-NEGATIVE ALPHAIBE  3ØØ
C         AND X. A COMBINATION OF THE POWER SERIES, THE ASYMPTOTIC    IBE  31Ø
C         EXPANSION FOR X TO INFINITY, AND THE UNIFORM ASYMPTOTIC     IBE  32Ø
C         EXPANSION FOR NU TO INFINITY ARE APPLIED OVER SUBDIVISIONS OFIBE  33Ø
C         THE (NU,X) PLANE. FOR VALUES NOT COVERED BY ONE OF THESE    IBE  34Ø
C         FORMULAE, THE ORDER IS INCREMENTED BY AN INTEGER SO THAT ONE IBE  35Ø
C         OF THESE FORMULAE APPLY. BACKWARD RECURSION IS USED TO REDUCEIBE  36Ø
C         ORDERS BY INTEGER VALUES. THE ASYMPTOTIC EXPANSION FOR X TO IBE  37Ø
C         INFINITY IS USED ONLY WHEN THE ENTIRE SEQUENCE (SPECIFICALLY IBE  38Ø
C         THE LAST MEMBER) LIES WITHIN THE REGION COVERED BY THE      IBE  39Ø
C         EXPANSION. LEADING TERMS OF THESE EXPANSIONS ARE USED TO TEST IBE  4ØØ
C         FOR OVER OR UNDERFLOW WHERE APPROPRIATE. IF A SEQUENCE IS   IBE  41Ø
C         REQUESTED AND THE LAST MEMBER WOULD UNDERFLOW, THE RESULT IS IBE  42Ø
C         SET TO ZERO AND THE NEXT LOWER ORDER TRIED, ETC., UNTIL A   IBE  43Ø
C         MEMBER COMES ON SCALE OR ALL ARE SET TO ZERO. AN OVERFLOW   IBE  44Ø
C         CANNOT OCCUR WITH SCALING. IBESS CALLS FUNCTION GAMLN.      IBE  45Ø
C                                                                     IBE  46Ø
C     DESCRIPTION OF ARGUMENTS                                        IBE  47Ø
C                                                                     IBE  48Ø
C         INPUT                                                       IBE  49Ø
C           KODE   - A PARAMETER TO INDICATE THE SCALING OPTION       IBE  5ØØ
C                    KODE=1 RETURNS                                   IBE  51Ø
C                           Y(K)=        I/SUB(ALPHA+K-1)/(X),       IBE  52Ø
C                                K=1,...,N                            IBE  53Ø
C                    KODE=2 RETURNS                                   IBE  54Ø
C                           Y(K)=EXP(-X)*I/SUB(ALPHA+K-1)/(X),       IBE  55Ø
C                                K=1,...,N                            IBE  56Ø
C           ALPHA  - ORDER OF FIRST MEMBER OF THE SEQUENCE, ALPHA.GE.Ø IBE  57Ø
C           N      - NUMBER OF MEMBERS IN THE SEQUENCE, N.GE.1        IBE  58Ø
C           X      - X.GE.Ø                                           IBE  59Ø
C                                                                     IBE  6ØØ
C         OUTPUT                                                      IBE  61Ø
C           Y      - A VECTOR WHOSE FIRST N COMPONENTS CONTAIN        IBE  62Ø
C                    VALUES FOR I/SUB(ALPHA+K-1)/(X) OR SCALED        IBE  63Ø
C                    VALUES FOR EXP(-X)*I/SUB(ALPHA+K-1)/(X),         IBE  64Ø
C                    K=1,...,N DEPENDING ON KODE                      IBE  65Ø
C                                                                     IBE  66Ø
C     ERROR CONDITIONS                                                IBE  67Ø
```

```
C         IMPROPER INPUT ARGUMENTS - A FATAL ERROR                          IBE  68Ø
C         OVERFLOW WITH KODE=1 - A FATAL ERROR                              IBE  69Ø
C         UNDERFLOW - A NON-FATAL ERROR                                     IBE  7ØØ
C                                                                           IBE  71Ø
      DOUBLE PRECISION DX, TRX, DTM, DFN                                    IBE  72Ø
      DIMENSION Y(1), TEMP(3)                                               IBE  73Ø
      DIMENSION C(11,1Ø)                                                    IBE  74Ø
      DIMENSION C1(11,8), C2(11,2)                                          IBE  75Ø
      EQUIVALENCE (C(1,1),C1(1,1))                                          IBE  76Ø
      EQUIVALENCE (C(1,9),C2(1,1))                                          IBE  77Ø
      DATA ELIM,TOL        /        667.        ,       1.E-15          /IBE  78Ø
      DATA RTPI,RTTPI      / 1.5915494309189 5E-Ø1, 3.9894228Ø4Ø1433E-Ø1/IBE  79Ø
      DATA CE              / 3.4538776394ØØØØE+Ø1/                          IBE  8ØØ
      DATA INLIM           /          8Ø         /                          IBE  81Ø
      DATA C1              /-2.Ø8333333333333E-Ø1, 1.25ØØØØØØØØØØØØE-Ø1,IBE  82Ø
     1         9(Ø.)        , 3.342Ø1388888889E-Ø1,-4.Ø1Ø41666666667E-Ø1,IBE  83Ø
     2 7.Ø3125ØØØØØØØØE-Ø2,          8(Ø.)        ,-1.Ø2581259645Ø62E+ØØ,IBE  84Ø
     3 1.84646267361111E+ØØ,-8.9121Ø9375ØØØØE-Ø1, 7.32421875ØØØØØE-Ø2,IBE  85Ø
     4         7(Ø.)        , 4.66958442342625E+ØØ,-1.12Ø7ØØ2616223ØE+Ø1,IBE  86Ø
     5 8.78912353515625E+ØØ,-2.364Ø869146Ø625ØE+ØØ, 1.12152Ø996Ø9375E-Ø1,IBE  87Ø
     6         6(Ø.)        ,-2.8212Ø725582ØØ2E+Ø1, 8.463621767460Ø7E+Ø1,IBE  88Ø
     7-9.18182415432400E+Ø1, 4.25349987453885E+Ø1,-7.36879435947963E+ØØ,IBE  89Ø
     8 2.271Ø8ØØ17Ø8984E-Ø1,          5(Ø.)        , 2.1257Ø13ØØ39217E+Ø2,IBE  9ØØ
     9-7.65252468141182E+Ø2, 1.Ø5999Ø4525280ØE+Ø3,-6.99579627376133E+Ø2,IBE  91Ø
     A 2.1819Ø511744212E+Ø2,-2.649143Ø4869516E+Ø1, 5.725Ø142Ø974731E-Ø1,IBE  92Ø
     B         4(Ø.)        ,-1.919457662318 41E+Ø3, 8.Ø6172218173731E+Ø3,IBE  93Ø
     C-1.3586550ØØ64341E+Ø4, 1.16553933368645E+Ø4,-5.3Ø564697861340E+Ø3,IBE  94Ø
     D 1.2ØØ9Ø291321635E+Ø3,-1.Ø8Ø9Ø919788395E+Ø2, 1.7277275Ø258446E+ØØ,IBE  95Ø
     E         3(Ø.)        , 2.Ø2Ø4291330 9661E+Ø4,-9.698Ø5983886375E+Ø4,IBE  96Ø
     F 1.92547ØØ1232532E+Ø5,-2.Ø34ØØ17728Ø416E+Ø5, 1.222ØØ464983Ø17E+Ø5,IBE  97Ø
     G-4.11926549688976E+Ø4, 7.1Ø95143Ø248936E+Ø3,-4.939153Ø4773Ø88E+Ø2,IBE  98Ø
     H 6.Ø74Ø42ØØ127348E+ØØ,          2(Ø.)        /                     IBE  99Ø
      DATA C2             /-2.4291918790Ø551E+Ø5, 1.31176361466298E+Ø6,IBE 1ØØØ
     1-2.998Ø1591853811E+Ø6, 3.7632712976564ØE+Ø6,-2.81356322658653E+Ø6,IBE 1Ø1Ø
     2 1.26836527332162E+Ø6,-3.3164517248456 4E+Ø5, 4.52187689813627E+Ø4,IBE 1Ø2Ø
     3-2.499830481811 21E+Ø3, 2.4380529699556 1E+Ø1,          1(Ø.)       ,IBE 1Ø3Ø
     4 3.2844698530720 4E+Ø6,-1.97Ø68191184322E+Ø7, 5.Ø9526Ø24926646E+Ø7,IBE 1Ø4Ø
     5-7.41Ø51482115327E+Ø7, 6.6344512274729ØE+Ø7,-3.756717666Ø7634E+Ø7,IBE 1Ø5Ø
     6 1.32887671664218E+Ø7,-2.7856181280 8645E+Ø6, 3.Ø8186Ø4612662E+Ø5,IBE 1Ø6Ø
     7-1.3886Ø897537170E+Ø4, 1.1ØØ1714Ø269247E+Ø2/                        IBE 1Ø7Ø
C TEST INPUT ARGUMENTS                                                      IBE 1Ø8Ø
      KT = 1                                                                IBE 1Ø9Ø
C TEST INPUT ARGUMENTS                                                      IBE 11ØØ
      IF (N-1) 58Ø, 1Ø, 2Ø                                                 IBE 111Ø
   1Ø KT = 2                                                                IBE 112Ø
   2Ø NN = N                                                                IBE 113Ø
      IF (KODE.LT.1 .OR. KODE.GT.2) GO TO 56Ø                               IBE 114Ø
      IF (X) 59Ø, 3Ø, 8Ø                                                   IBE 115Ø
   3Ø IF (ALPHA) 57Ø, 4Ø, 5Ø                                               IBE 116Ø
   4Ø Y(1) = 1.                                                             IBE 117Ø
      IF (N.EQ.1) RETURN                                                    IBE 118Ø
      I1 = 2                                                                IBE 119Ø
      GO TO 6Ø                                                              IBE 12ØØ
   5Ø I1 = 1                                                                IBE 121Ø
   6Ø DO 7Ø I=I1,N                                                          IBE 122Ø
        Y(I) = Ø.                                                           IBE 123Ø
   7Ø CONTINUE                                                              IBE 124Ø
      RETURN                                                                IBE 125Ø
   8Ø CONTINUE                                                              IBE 126Ø
      IF (ALPHA.LT.Ø.) GO TO 57Ø                                            IBE 127Ø
      DFN = DBLE(FLOAT(N)) + DBLE(ALPHA) - 1.D+Ø                            IBE 128Ø
      FNU = DFN                                                             IBE 129Ø
      IN = Ø                                                                IBE 13ØØ
      XO2 = X*.5                                                            IBE 131Ø
      SXO2 = XO2*XO2                                                        IBE 132Ø
      ETX = KODE - 1                                                        IBE 133Ø
      SX = ETX*X                                                            IBE 134Ø
C DECISION TREE FOR REGION WHERE SERIES, ASYMPTOTIC EXPANSION FOR X         IBE 135Ø
C TO INFINITY AND ASYMPTOTIC EXPANSION FOR NU TO INFINITY ARE               IBE 136Ø
C APPLIED.                                                                  IBE 137Ø
      IF (SXO2.LE.(FNU+1.)) GO TO 9Ø                                        IBE 138Ø
      IF (X.LE.12.) GO TO 11Ø                                               IBE 139Ø
      FN = Ø.55*FNU*FNU                                                     IBE 14ØØ
      FN = AMAX1(17.,FN)                                                    IBE 141Ø
      IF (X.GE.FN) GO TO 43Ø                                                IBE 142Ø
      NS = AMAX1(36.-FNU,Ø.)                                                IBE 143Ø
```

```
      DFN = DFN + DBLE(FLOAT(NS))                                    IBE 144Ø
      FN = DFN                                                       IBE 145Ø
      IS = KT                                                        IBE 146Ø
      KM = N - 1 + NS                                                IBE 147Ø
      IF (KM.GT.Ø) IS = 3                                            IBE 148Ø
      GO TO 12Ø                                                      IBE 149Ø
   9Ø FN = FNU                                                       IBE 15ØØ
      FNP1 = FN + 1.                                                 IBE 151Ø
      XO2L = ALOG(XO2)                                               IBE 152Ø
      IS = KT                                                        IBE 153Ø
      IF (X.LE.Ø.5) GO TO 23Ø                                        IBE 154Ø
      NS = Ø                                                         IBE 155Ø
  1ØØ DFN = DFN + DBLE(FLOAT(NS))                                    IBE 156Ø
      FN = DFN                                                       IBE 157Ø
      FNP1 = FN + 1.                                                 IBE 158Ø
      IS = KT                                                        IBE 159Ø
      IF (N-1+NS.GT.Ø) IS = 3                                        IBE 16ØØ
      GO TO 23Ø                                                      IBE 161Ø
  11Ø XO2L = ALOG(XO2)                                               IBE 162Ø
      NS = SXO2 - FNU                                                IBE 163Ø
      GO TO 1ØØ                                                      IBE 164Ø
  12Ø CONTINUE                                                       IBE 165Ø
C OVERFLOW TEST ON UNIFORM ASYMPTOTIC EXPANSION                      IBE 166Ø
      IF (KODE.EQ.2) GO TO 13Ø                                       IBE 167Ø
      IF (ALPHA.LT.1.) GO TO 15Ø                                     IBE 168Ø
      Z = X/ALPHA                                                    IBE 169Ø
      RA = SQRT(1.+Z*Z)                                              IBE 17ØØ
      GLN = ALOG((1.+RA)/Z)                                          IBE 171Ø
      T = RA*(1.-ETX) + ETX/(Z+RA)                                   IBE 172Ø
      ARG = ALPHA*(T-GLN)                                            IBE 173Ø
      IF (ARG.GT.ELIM) GO TO 6ØØ                                     IBE 174Ø
      IF (KM.EQ.Ø) GO TO 14Ø                                         IBE 175Ø
  13Ø CONTINUE                                                       IBE 176Ø
C UNDERFLOW TEST ON UNIFORM ASYMPTOTIC EXPANSION                     IBE 177Ø
      Z = X/FN                                                       IBE 178Ø
      RA = SQRT(1.+Z*Z)                                              IBE 179Ø
      GLN = ALOG((1.+RA)/Z)                                          IBE 18ØØ
      T = RA*(1.-ETX) + ETX/(Z+RA)                                   IBE 181Ø
      ARG = FN*(T-GLN)                                               IBE 182Ø
  14Ø IF (ARG.LT.-ELIM) GO TO 28Ø                                    IBE 183Ø
      GO TO 19Ø                                                      IBE 184Ø
  15Ø IF (X.GT.ELIM) GO TO 6ØØ                                      IBE 185Ø
      GO TO 13Ø                                                      IBE 186Ø
C UNIFORM ASYMPTOTIC EXPANSION FOR NU TO INFINITY                    IBE 187Ø
  16Ø IF (KM.NE.Ø) GO TO 17Ø                                         IBE 188Ø
      Y(1) = TEMP(3)                                                 IBE 189Ø
      RETURN                                                         IBE 19ØØ
  17Ø TEMP(1) = TEMP(3)                                              IBE 191Ø
      IN = NS                                                        IBE 192Ø
      KT = 1                                                         IBE 193Ø
  18Ø CONTINUE                                                       IBE 194Ø
      IS = 2                                                         IBE 195Ø
      DFN = DFN - 1.D+Ø                                              IBE 196Ø
      FN = DFN                                                       IBE 197Ø
      Z = X/FN                                                       IBE 198Ø
      RA = SQRT(1.+Z*Z)                                              IBE 199Ø
      GLN = ALOG((1.+RA)/Z)                                          IBE 2ØØØ
      T = RA*(1.-ETX) + ETX/(Z+RA)                                   IBE 2Ø1Ø
      ARG = FN*(T-GLN)                                               IBE 2Ø2Ø
  19Ø COEF = EXP(ARG)                                                IBE 2Ø3Ø
      T = 1./RA                                                      IBE 2Ø4Ø
      T2 = T*T                                                       IBE 2Ø5Ø
      T = T/FN                                                       IBE 2Ø6Ø
      S2 = 1.                                                        IBE 2Ø7Ø
      AP = 1.                                                        IBE 2Ø8Ø
      DO 21Ø K=1,1Ø                                                  IBE 2Ø9Ø
        KP1 = K + 1                                                  IBE 21ØØ
        S1 = C(1,K)                                                  IBE 211Ø
        DO 2ØØ J=2,KP1                                               IBE 212Ø
          S1 = S1*T2 + C(J,K)                                        IBE 213Ø
  2ØØ   CONTINUE                                                     IBE 214Ø
        AP = AP*T                                                    IBE 215Ø
        AK = AP*S1                                                   IBE 216Ø
        S2 = S2 + AK                                                 IBE 217Ø
        IF (ABS(AK).LT.TOL) GO TO 22Ø                                IBE 218Ø
  21Ø CONTINUE                                                       IBE 219Ø
```

```
  22Ø CONTINUE                                                          IBE 22ØØ
      TEMP(IS) = SQRT(T*RTPI)*COEF*S2                                   IBE 221Ø
      GO TO (18Ø, 35Ø, 5ØØ), IS                                         IBE 222Ø
C SERIES FOR (X/2)**2.LE.NU+1                                           IBE 223Ø
  23Ø CONTINUE                                                          IBE 224Ø
      GLN = GAMLN(FNP1)                                                 IBE 225Ø
      ARG = FN*XO2L - GLN - SX                                          IBE 226Ø
      IF (ARG.LT.-ELIM) GO TO 3ØØ                                       IBE 227Ø
      EARG = EXP(ARG)                                                   IBE 228Ø
  24Ø CONTINUE                                                          IBE 229Ø
      S = 1.                                                            IBE 23ØØ
      AK = 3.                                                           IBE 231Ø
      T2 = 1.                                                           IBE 232Ø
      T = 1.                                                            IBE 233Ø
      S1 = FN                                                           IBE 234Ø
      DO 25Ø K=1,17                                                     IBE 235Ø
        S2 = T2 + S1                                                    IBE 236Ø
        T = T*SXO2/S2                                                   IBE 237Ø
        S = S + T                                                       IBE 238Ø
        IF (ABS(T).LT.TOL) GO TO 26Ø                                    IBE 239Ø
        T2 = T2 + AK                                                    IBE 24ØØ
        AK = AK + 2.                                                    IBE 241Ø
        S1 = S1 + FN                                                    IBE 242Ø
  25Ø CONTINUE                                                          IBE 243Ø
  26Ø CONTINUE                                                          IBE 244Ø
      TEMP(IS) = S*EARG                                                 IBE 245Ø
      GO TO (27Ø, 35Ø, 49Ø), IS                                         IBE 246Ø
  27Ø EARG = EARG*FN/XO2                                                IBE 247Ø
      DFN = DFN - 1.D+Ø                                                 IBE 248Ø
      FN = DFN                                                          IBE 249Ø
      IS = 2                                                            IBE 25ØØ
      GO TO 24Ø                                                         IBE 251Ø
C SET UNDERFLOW VALUE AND UPDATE PARAMETERS                             IBE 252Ø
  28Ø Y(NN) = Ø.                                                        IBE 253Ø
      NN = NN - 1                                                       IBE 254Ø
      DFN = DFN - 1.D+Ø                                                 IBE 255Ø
      FN = DFN                                                          IBE 256Ø
      IF (NN-1) 34Ø, 29Ø, 13Ø                                           IBE 257Ø
  29Ø KT = 2                                                            IBE 258Ø
      IS = 2                                                            IBE 259Ø
      GO TO 13Ø                                                         IBE 26ØØ
  3ØØ Y(NN) = Ø.                                                        IBE 261Ø
      NN = NN - 1                                                       IBE 262Ø
      FNP1 = FN                                                         IBE 263Ø
      DFN = DFN - 1.D+Ø                                                 IBE 264Ø
      FN = DFN                                                          IBE 265Ø
      IF (NN-1) 34Ø, 31Ø, 32Ø                                           IBE 266Ø
  31Ø KT = 2                                                            IBE 267Ø
      IS = 2                                                            IBE 268Ø
  32Ø IF (SXO2.LE.FNP1) GO TO 33Ø                                       IBE 269Ø
      GO TO 13Ø                                                         IBE 27ØØ
  33Ø ARG = ARG - XO2L + ALOG(FNP1)                                     IBE 271Ø
      IF (ARG.LT.-ELIM) GO TO 3ØØ                                       IBE 272Ø
      GO TO 23Ø                                                         IBE 273Ø
  34Ø NZ = N - NN                                                       IBE 274Ø
      PRINT 99994, NZ, KODE, ALPHA, N, X                                IBE 275Ø
      RETURN                                                            IBE 276Ø
C BACKWARD RECURSION SECTION                                            IBE 277Ø
  35Ø CONTINUE                                                          IBE 278Ø
      NZ = N - NN                                                       IBE 279Ø
      IF (NZ.NE.Ø) PRINT 99994, NZ, KODE, ALPHA, N, X                   IBE 28ØØ
  36Ø GO TO (37Ø, 42Ø), KT                                              IBE 281Ø
  37Ø CONTINUE                                                          IBE 282Ø
      S1 = TEMP(1)                                                      IBE 283Ø
      S2 = TEMP(2)                                                      IBE 284Ø
      DX = X                                                            IBE 285Ø
      TRX = 2.D+Ø/DX                                                    IBE 286Ø
      DTM = DFN*TRX                                                     IBE 287Ø
      TM = DTM                                                          IBE 288Ø
      IF (IN.EQ.Ø) GO TO 39Ø                                            IBE 289Ø
C BACKWARD RECUR TO INDEX ALPHA+NN-1                                    IBE 29ØØ
      DO 38Ø I=1,IN                                                     IBE 291Ø
        S = S2                                                          IBE 292Ø
        S2 = TM*S2 + S1                                                 IBE 293Ø
        S1 = S                                                          IBE 294Ø
        DTM = DTM - TRX                                                 IBE 295Ø
```

```
        TM = DTM                                                        IBE 296Ø
  38Ø CONTINUE                                                          IBE 297Ø
      Y(NN) = S1                                                        IBE 298Ø
      IF (NN.EQ.1) RETURN                                               IBE 299Ø
      Y(NN-1) = S2                                                      IBE 3ØØØ
      IF (NN.EQ.2) RETURN                                               IBE 3Ø1Ø
      GO TO 4ØØ                                                         IBE 3Ø2Ø
  39Ø CONTINUE                                                          IBE 3Ø3Ø
C BACKWARD RECUR FROM INDEX ALPHA+NN-1 TO ALPHA                         IBE 3Ø4Ø
      Y(NN) = S1                                                        IBE 3Ø5Ø
      Y(NN-1) = S2                                                      IBE 3Ø6Ø
      IF (NN.EQ.2) RETURN                                               IBE 3Ø7Ø
  4ØØ K = NN + 1                                                        IBE 3Ø8Ø
      DO 41Ø I=3,NN                                                     IBE 3Ø9Ø
        K = K - 1                                                       IBE 31ØØ
        Y(K-2) = TM*Y(K-1) + Y(K)                                       IBE 311Ø
        DTM = DTM - TRX                                                 IBE 312Ø
        TM = DTM                                                        IBE 313Ø
  41Ø CONTINUE                                                          IBE 314Ø
      RETURN                                                            IBE 315Ø
  42Ø Y(1) = TEMP(2)                                                    IBE 316Ø
      RETURN                                                            IBE 317Ø
C ASYMPTOTIC EXPANSION FOR X TO INFINITY                                IBE 318Ø
  43Ø CONTINUE                                                          IBE 319Ø
      EARG = RTTPI/SQRT(X)                                              IBE 32ØØ
      IF (KODE.EQ.2) GO TO 44Ø                                          IBE 321Ø
      IF (X.GT.ELIM) GO TO 6ØØ                                          IBE 322Ø
      EARG = EARG*EXP(X)                                                IBE 323Ø
  44Ø ETX = 8.*X                                                        IBE 324Ø
      IS = KT                                                           IBE 325Ø
      IN = Ø                                                            IBE 326Ø
  45Ø DX = DFN + DFN                                                    IBE 327Ø
      DTM = DX*DX                                                       IBE 328Ø
      S1 = ETX                                                          IBE 329Ø
      TRX = S1                                                          IBE 33ØØ
      DX = -(DTM-1.D+Ø)/TRX                                             IBE 331Ø
      T = DX                                                            IBE 332Ø
      TRX = 1.D+Ø + DX                                                  IBE 333Ø
      S = TRX                                                           IBE 334Ø
      S2 = 1.                                                           IBE 335Ø
      AK = 8.                                                           IBE 336Ø
      DO 46Ø K=1,25                                                     IBE 337Ø
        S1 = S1 + ETX                                                   IBE 338Ø
        S2 = S2 + AK                                                    IBE 339Ø
        DX = S2                                                         IBE 34ØØ
        TRX = DTM - DX                                                  IBE 341Ø
        AP = TRX                                                        IBE 342Ø
        T = -T*AP/S1                                                    IBE 343Ø
        S = S + T                                                       IBE 344Ø
        IF (ABS(T).LE.TOL) GO TO 47Ø                                    IBE 345Ø
        AK = AK + 8.                                                    IBE 346Ø
  46Ø CONTINUE                                                          IBE 347Ø
  47Ø TEMP(IS) = S*EARG                                                 IBE 348Ø
      GO TO (48Ø, 36Ø), IS                                              IBE 349Ø
  48Ø IS = 2                                                            IBE 35ØØ
      DFN = DFN - 1.D+Ø                                                 IBE 351Ø
      GO TO 45Ø                                                         IBE 352Ø
C BACKWARD RECURSION WITH NORMALIZATION BY                              IBE 353Ø
C ASYMPTOTIC EXPANSION FOR NU TO INFINITY OR POWER SERIES.              IBE 354Ø
  49Ø CONTINUE                                                          IBE 355Ø
C COMPUTATION OF LAST ORDER FOR SERIES NORMALIZATION                    IBE 356Ø
      KM = AMAX1(3.-FN,Ø.)                                              IBE 357Ø
      TFN = FN + FLOAT(KM)                                              IBE 358Ø
      TA = (GLN+TFN-Ø.9189385332-Ø.Ø833333333/TFN)/(TFN+Ø.5)            IBE 359Ø
      TA = XO2L - TA                                                    IBE 36ØØ
      TB = -(1.-1./TFN)/TFN                                             IBE 361Ø
      IN = CE/(-TA+SQRT(TA*TA-CE*TB)) + 1.5                             IBE 362Ø
      IN = IN + KM                                                      IBE 363Ø
      GO TO 51Ø                                                         IBE 364Ø
  5ØØ CONTINUE                                                          IBE 365Ø
C COMPUTATION OF LAST ORDER FOR ASYMPTOTIC EXPANSION NORMALIZATION      IBE 366Ø
      IN = CE/(GLN+SQRT(GLN*GLN+T*CE)) + 1.5                            IBE 367Ø
      IF (IN.GT.INLIM) GO TO 16Ø                                        IBE 368Ø
  51Ø DX = FLOAT(IN)                                                    IBE 369Ø
      DTM = DFN + DX                                                    IBE 37ØØ
      DX = X                                                            IBE 371Ø
```

```
      TRX = 2.D+Ø/DX                                                    IBE 372Ø
      DTM = DTM*TRX                                                     IBE 373Ø
      TM = DTM                                                          IBE 374Ø
      TA = Ø.                                                           IBE 375Ø
      TB = TOL                                                          IBE 376Ø
      KK = 1                                                            IBE 377Ø
  52Ø CONTINUE                                                          IBE 378Ø
C BACKWARD RECUR UNINDEXED                                              IBE 379Ø
      DO 53Ø I=1,IN                                                     IBE 38ØØ
        S = TB                                                          IBE 381Ø
        TB = TM*TB + TA                                                 IBE 382Ø
        TA = S                                                          IBE 383Ø
        DTM = DTM - TRX                                                 IBE 384Ø
        TM = DTM                                                        IBE 385Ø
  53Ø CONTINUE                                                          IBE 386Ø
C NORMALIZATION                                                         IBE 387Ø
      IF (KK.NE.1) GO TO 54Ø                                            IBE 388Ø
      TA = (TA/TB)*TEMP(3)                                              IBE 389Ø
      TB = TEMP(3)                                                      IBE 39ØØ
      KK = 2                                                            IBE 391Ø
      IN = NS                                                           IBE 392Ø
      IF (NS.NE.Ø) GO TO 52Ø                                            IBE 393Ø
  54Ø Y(NN) = TB                                                        IBE 394Ø
      NZ = N - NN                                                       IBE 395Ø
      IF (NZ.NE.Ø) PRINT 99994, NZ, KODE, ALPHA, N, X                   IBE 396Ø
      IF (NN.EQ.1) RETURN                                               IBE 397Ø
      TB = TM*TB + TA                                                   IBE 398Ø
      K = NN - 1                                                        IBE 399Ø
      Y(K) = TB                                                         IBE 4ØØØ
      IF (NN.EQ.2) RETURN                                               IBE 4Ø1Ø
      DTM = DTM - TRX                                                   IBE 4Ø2Ø
      TM = DTM                                                          IBE 4Ø3Ø
      KM = K - 1                                                        IBE 4Ø4Ø
C BACKWARD RECUR INDEXED                                                IBE 4Ø5Ø
      DO 55Ø I=1,KM                                                     IBE 4Ø6Ø
        Y(K-1) = TM*Y(K) + Y(K+1)                                       IBE 4Ø7Ø
        DTM = DTM - TRX                                                 IBE 4Ø8Ø
        TM = DTM                                                        IBE 4Ø9Ø
        K = K - 1                                                       IBE 41ØØ
  55Ø CONTINUE                                                          IBE 411Ø
      RETURN                                                            IBE 412Ø
  56Ø PRINT 99999, KODE, ALPHA, N, X                                    IBE 413Ø
      STOP                                                              IBE 414Ø
  57Ø PRINT 99998, KODE, ALPHA, N, X                                    IBE 415Ø
      STOP                                                              IBE 416Ø
  58Ø PRINT 99997, KODE, ALPHA, N, X                                    IBE 417Ø
      STOP                                                              IBE 418Ø
  59Ø PRINT 99996, KODE, ALPHA, N, X                                    IBE 419Ø
      STOP                                                              IBE 42ØØ
  6ØØ PRINT 99995, KODE, ALPHA, N, X                                    IBE 421Ø
      STOP                                                              IBE 422Ø
99999 FORMAT (51HØIBESS CALLED WITH SCALING OPTION, KODE, NOT 1 OR 2    IBE 423Ø
     * /6H KODE=,I2,7H ALPHA=,E25.14,3H N=,I6,3H X=,E25.14)              IBE 424Ø
99998 FORMAT (51HØIBESS CALLED WITH THE ORDER, ALPHA, LESS THAN ZERO    IBE 425Ø
     * /6H KODE=,I2,7H ALPHA=,E25.14,3H N=,I6,3H X=,E25.14)              IBE 426Ø
99997 FORMAT (34HØIBESS CALLED WITH N LESS THAN ONE/6H KODE=,           IBE 427Ø
     * I2,7H ALPHA=,E25.14,3H N=,I6,3H X=,E25.14)                       IBE 428Ø
99996 FORMAT (35HØIBESS CALLED WITH X LESS THAN ZERO/6H KODE=,          IBE 429Ø
     * I2,7H ALPHA=,E25.14,3H N=,I6,3H X=,E25.14)                       IBE 43ØØ
99995 FORMAT (42HØOVERFLOW IN IBESS, X TOO LARGE FOR KODE=1/6H KODE=,   IBE 431Ø
     * I2,7H ALPHA=,E25.14,3H N=,I6,3H X=,E25.14)                       IBE 432Ø
99994 FORMAT (25HØUNDERFLOW IN IBESS, LAST,I6,2ØH VALUE(S) OF Y ARRAY,  IBE 433Ø
     * 17H WERE SET TO ZERO/6H KODE=,I2,7H ALPHA=,E25.14,3H N=,I6,      IBE 434Ø
     * 3H X=,E25.14)                                                    IBE 435Ø
      END                                                               IBE 436Ø


      SUBROUTINE JAIRY(X, RX, C, AI, DAI)                               JAI   1Ø
C                                                                       JAI   2Ø
C     CDC 66ØØ ROUTINE                                                  JAI   3Ø
C     1-2-74                                                            JAI   4Ø
C                                                                       JAI   5Ø
C                 JAIRY COMPUTES THE AIRY FUNCTION AI(X)                JAI   6Ø
C                  AND ITS DERIVATIVE DAI(X) FOR JBESS                  JAI   7Ø
C                                 INPUT                                 JAI   8Ø
```

```
C                                                                       JAI   9Ø
C        X - ARGUMENT, COMPUTED BY JBESS, X.LE.(ELIM2*1.5)**(2./3.)     JAI   1ØØ
C       RX - RX=SQRT(ABS(X)), COMPUTED BY JBESS                         JAI   11Ø
C        C - C=2.*(ABS(X)**1.5)/3., COMPUTED BY JBESS                   JAI   12Ø
C                                                                       JAI   13Ø
C                              OUTPUT                                   JAI   14Ø
C                                                                       JAI   15Ø
C       AI - VALUE OF FUNCTION AI(X)                                    JAI   16Ø
C      DAI - VALUE OF THE DERIVATIVE DAI(X)                             JAI   17Ø
C                                                                       JAI   18Ø
C                              WRITTEN BY                               JAI   19Ø
C                                                                       JAI   2ØØ
C                              D. E. AMOS                               JAI   21Ø
C                             S. L. DANIEL                              JAI   22Ø
C                             M. K. WESTON                              JAI   23Ø
C                                                                       JAI   24Ø
      DIMENSION AJP(19), AJN(19), A(15), B(15)                          JAI   25Ø
      DIMENSION AK1(14), AK2(23), AK3(14)                               JAI   26Ø
      DIMENSION DAJP(19), DAJN(19), DA(15), DB(15)                      JAI   27Ø
      DIMENSION DAK1(14), DAK2(24), DAK3(14)                            JAI   28Ø
      DATA N1, N2, N3, N4 /14,23,19,15/                                 JAI   29Ø
      DATA M1, M2, M3, M4 /12,21,17,13/                                 JAI   3ØØ
      DATA FPI12,CON2,CON3,CON4,CON5            / 1.3Ø899693899575E+ØØ,JAI   31Ø
     1 5.Ø3154716196777E+ØØ, 3.8ØØØ4589867293E-Ø1, 8.33333333333333E-Ø1,JAI   32Ø
     2 8.66Ø254Ø3784439E-Ø1/                                            JAI   33Ø
      DATA  AK1             / 2.2Ø423Ø9Ø987793E-Ø1,-1.2529Ø2427877ØØE-Ø1,JAI   34Ø
     1 1.Ø3881163359194E-Ø2, 8.22844152ØØ6343E-Ø4,-2.34614345891226E-Ø4,JAI   35Ø
     2 1.6382428Ø172116E-Ø5, 3.Ø69Ø2589573189E-Ø7,-1.29621999359332E-Ø7,JAI   36Ø
     3 8.229Ø8158823668E-Ø9, 1.53963968623298E-11,-3.39165465615682E-11,JAI   37Ø
     4 2.Ø3253257423626E-12,-1.1Ø6795546Ø97884E-14,-5.16169497785Ø8ØE-15/JAI   38Ø
      DATA  AK2             / 2.7436615Ø869598E-Ø1, 5.3979Ø969736ØØ3E-Ø3,JAI   39Ø
     1-1.5733922Ø62119ØE-Ø3, 4.2742752824875ØE-Ø4,-1.12124917399925E-Ø4,JAI   4ØØ
     2 2.88763171318ØØ4E-Ø5,-7.368Ø422537Ø554E-Ø6, 1.8729Ø2Ø9741Ø24E-Ø6,JAI   41Ø
     3-4.75892793962291E-Ø7, 1.2113Ø416955ØØ9E-Ø7,-3.Ø924537427Ø614E-Ø8,JAI   42Ø
     4 7.92454705282654E-Ø9,-2.Ø39Ø2447167914E-Ø9, 5.26863Ø56595742E-1Ø,JAI   43Ø
     5-1.367Ø4767639569E-1Ø, 3.56141Ø39Ø137Ø8E-11,-9.31388296548430E-12,JAI   44Ø
     6 2.4446445Ø473635E-12,-6.4384Ø26199Ø955E-13, 1.7Ø1Ø6Ø3Ø559349E-13,JAI   45Ø
     7-4.5Ø76Ø1Ø45Ø3281E-14, 1.19774799164811E-14,-3.19Ø77Ø4Ø865Ø66E-15/JAI   46Ø
      DATA  AK3             / 2.8Ø27144734Ø791E-Ø1,-1.78127Ø42844379E-Ø3,JAI   47Ø
     1 4.Ø3422579628999E-Ø5,-1.6324996526ØØ3E-Ø6, 9.21181482476768E-Ø8,JAI   48Ø
     2-6.522943Ø229155E-Ø9, 5.471384Ø4576546E-1Ø,-5.244Ø8251ØØ26ØE-11,JAI   49Ø
     3 5.6Ø4779Ø41172Ø9E-12,-6.56375244639313E-13, 8.31285761966247E-14,JAI   5ØØ
     4-1.127Ø5134691Ø63E-14, 1.62267976598129E-15,-2.4648Ø324312426E-16/JAI   51Ø
      DATA  AJP             / 7.78952966437581E-Ø2,-1.8435636345681E-Ø1,JAI   52Ø
     1 3.Ø14126Ø5216174E-Ø2, 3.Ø5342724277Ø8E-Ø2,-4.954247Ø2513Ø79E-Ø3,JAI   53Ø
     2-1.72749552563952E-Ø3, 2.4313763783919ØE-Ø4, 5.Ø4564777517Ø82E-Ø5,JAI   54Ø
     3-6.1631658269520Ø8E-Ø6,-9.Ø3986745510768E-Ø7, 9.7Ø243778355884E-Ø8,JAI   55Ø
     4 1.Ø96394533Ø52Ø5E-Ø8,-1.Ø4716330588766E-Ø9,-9.6Ø359441344646E-11,JAI   56Ø
     5 8.25358789454134E-12, 6.36123439Ø18768E-13,-4.966296141160152-14,JAI   57Ø
     6-3.2981Ø288929615E-15, 2.3579852Ø311Ø4E-16/                       JAI   58Ø
      DATA  AJN             / 3.8Ø497887617242E-Ø2,-2.45319541845546E-Ø1,JAI   59Ø
     1 1.6582Ø6237Ø2696E-Ø1, 7.4933ØØ45818789E-Ø2,-2.6347628810664 1E-Ø2,JAI   6ØØ
     2-5.9253559730498 1E-Ø3, 1.447444Ø9589804E-Ø3, 2.18311831322215E-Ø4,JAI   61Ø
     3-4.1Ø662Ø7768Ø3Ø4E-Ø5,-4.66874994171766E-Ø6, 7.152188Ø727716ØE-Ø7,JAI   62Ø
     4 6.5296477Ø854633E-Ø8,-8.44284Ø27565946E-Ø9,-6.44186158976978E-1Ø,JAI   63Ø
     5 7.2Ø8Ø22865Ø5285E-11, 4.72465431717846E-12,-4.66Ø22632547Ø45E-13,JAI   64Ø
     6-2.677627 1Ø389189E-14, 2.3616131657ØØ19E-15/                     JAI   65Ø
      DATA   A              / 4.9Ø275424742791E-Ø1, 1.57647277946Ø4E-Ø3,JAI   66Ø
     1-9.66195963140306E-Ø5, 1.35916Ø8Ø268815E-Ø7, 2.98157342654859E-Ø7,JAI   67Ø
     2-1.86824767559979E-Ø8,-1.Ø3685737667141E-Ø9, 3.2866Ø818434328E-1Ø,JAI   68Ø
     3-2.57Ø9141Ø63278Ø6E-11,-2.3235765530ØØ677E-12, 9.57523279Ø48255E-13,JAI   69Ø
     4-1.2Ø34Ø828Ø49719E-13,-2.9Ø9Ø771677Ø715E-15, 4.5565645458Ø149E-15,JAI   7ØØ
     5-9.99ØØ387481Ø259E-16/                                            JAI   71Ø
      DATA   B              / 2.785935528Ø3Ø79E-Ø1,-3.52915691882584E-Ø3,JAI   72Ø
     1-2.31149677384994E-Ø5, 4.7131784226356ØE-Ø6,-1.124159Ø7931333E-Ø7,JAI   73Ø
     2-2.ØØ1ØØ3Ø1184339E-Ø8, 2.6Ø948Ø753Ø2193E-Ø9,-3.55Ø9813614Ø1216E-11,JAI   74Ø
     3-3.5Ø849978423875E-11, 5.83ØØ71879542Ø2E-12,-2.Ø4644828753326E-13,JAI   75Ø
     4-1.1Ø529179476742E-13, 2.87724778Ø38775E-14,-2.882Ø5111ØØ9939E-15,JAI   76Ø
     5-3.32656311696166E-16/                                            JAI   77Ø
      DATA N1D, N2D, N3D, N4D /14,24,19,15/                             JAI   78Ø
      DATA M1D, M2D, M3D, M4D /12,22,17,13/                             JAI   79Ø
      DATA  DAK1            / 2.Ø4567842307887E-Ø1,-6.6132273990566 4E-Ø2,JAI   8ØØ
     1-8.498458ØØ989287E-Ø3, 3.12183491556289E-Ø3,-2.7ØØ16489829432E-Ø4,JAI   81Ø
     2-6.35636298679387E-Ø6, 3.Ø2397712409Ø9ØE-Ø6,-2.183111953ØØ88E-Ø7,JAI   82Ø
     3-5.36194289332826E-1Ø, 1.13Ø98Ø3562231ØE-Ø9,-7.43Ø2383462973E-11,JAI   83Ø
     4 4.288Ø417Ø826891E-13, 2.2381Ø925754539E-13,-1.3914Ø135641182E-14/JAI   84Ø
```

```
      DATA  DAK2           / 2.93332343883230E-01,-8.06196784743112E-03,JAI  850
     1 2.42540172333140E-03,-6.82297548850235E-04, 1.85786427751181E-04,JAI  860
     2-4.97457447684059E-05, 1.32090681239497E-05,-3.49528240444943E-06,JAI  870
     3 9.24362451078835E-07,-2.44732671521867E-07, 6.49307837648910E-08,JAI  880
     4-1.72717621501538E-08, 4.60725763604656E-09,-1.23249055291550E-09,JAI  890
     5 3.30620409488102E-10,-8.89252099772401E-11, 2.39773319878298E-11,JAI  900
     6-6.48013921153450E-12, 1.75510132023731E-12,-4.76303829833637E-13,JAI  910
     7 1.29498241100810E-13,-3.52679622210430E-14, 9.62005151585923E-15,JAI  920
     8-2.62786914342292E-15/                                            JAI  930
      DATA  DAK3           / 2.84675828811349E-01, 2.53073072619080E-03,JAI  940
     1-4.83481130337976E-05, 1.84907283946343E-06,-1.01418491178576E-07,JAI  950
     2 7.05925634457153E-09,-5.85325291400382E-10, 5.56357688831339E-11,JAI  960
     3-5.90889094779500E-12, 6.88574353784436E-13,-8.68588256452194E-14,JAI  970
     4 1.17374762617213E-14,-1.68523146510923E-15, 2.55374773097056E-16/JAI  980
      DATA  DAJP           / 6.53219131311457E-02,-1.20262933688823E-01,JAI  990
     1 9.78010236263823E-03, 1.67948429230505E-02,-1.97146140182132E-03,JAI 1000
     2-8.45560295098867E-04, 9.42889620701976E-05, 2.25827860945475E-05,JAI 1010
     3-2.29067870915987E-06,-3.76343991136919E-07, 3.45663933559565E-08,JAI 1020
     4 4.29611332003007E-09,-3.58673691214989E-10,-3.57245881361895E-11,JAI 1030
     5 2.72696091066336E-12, 2.26120653095771E-13,-1.58763205238303E-14,JAI 1040
     6-1.12604374485125E-15, 7.31327529515367E-17/                      JAI 1050
      DATA  DAJN           / 1.08594539632967E-02, 8.53313194857091E-02,JAI 1060
     1-3.15277068113058E-01,-8.78420725294257E-02, 5.53251906976048E-02,JAI 1070
     2 9.41674060503241E-03,-3.32187026018996E-03,-4.11157343156826E-04,JAI 1080
     3 1.01297326891346E-04, 9.87633682208396E-06,-1.87312969812393E-06,JAI 1090
     4-1.50798500131468E-07, 2.32687669525394E-08, 1.59599917419225E-09,JAI 1100
     5-2.07665922668385E-10,-1.24103350500302E-11, 1.39631765331043E-12,JAI 1110
     6 7.39400971155740E-14,-7.32887475627500E-15/                      JAI 1120
      DATA   DA            / 4.91627321104601E-01, 3.11164930427489E-03,JAI 1130
     1 8.23140762854081E-05,-4.61769776172142E-06,-6.13158880534626E-08,JAI 1140
     2 2.87295804656520E-08,-1.81959715372117E-09,-1.44752826642035E-10,JAI 1150
     3 4.53724043420422E-11,-3.99655065847223E-12,-3.24089119830323E-13,JAI 1160
     4 1.62098952568741E-13,-2.40765247974057E-14, 1.69384811284491E-16,JAI 1170
     5 8.17900786477396E-16/                                            JAI 1180
      DATA   DB            /-2.77571356944231E-01, 4.44212833419920E-03,JAI 1190
     1-8.42328522190089E-05,-2.58040318418710E-06, 3.42389720217621E-07,JAI 1200
     2-6.24286894709776E-09,-2.36377836844577E-09, 3.16991042656673E-10,JAI 1210
     3-4.40995691658191E-12,-5.18674221093575E-12, 9.64874015137022E-13,JAI 1220
     4-4.90190576608710E-14,-1.77253430678112E-14, 5.55950610442662E-15,JAI 1230
     5-7.11793337579530E-16/                                            JAI 1240
      IF (X.LT.0.) GO TO 90                                             JAI 1250
      IF (C.GT.5.) GO TO 60                                             JAI 1260
      IF (X.GT.1.2) GO TO 30                                            JAI 1270
      T = (X+X-1.2)*CON4                                                JAI 1280
      TT = T + T                                                        JAI 1290
      J = N1                                                            JAI 1300
      F1 = AK1(J)                                                       JAI 1310
      F2 = 0.                                                           JAI 1320
      DO 10 I=1,M1                                                      JAI 1330
        J = J - 1                                                       JAI 1340
        TEMP1 = F1                                                      JAI 1350
        F1 = TT*F1 - F2 + AK1(J)                                        JAI 1360
        F2 = TEMP1                                                      JAI 1370
   10 CONTINUE                                                          JAI 1380
      AI = T*F1 - F2 + AK1(1)                                           JAI 1390
      J = N1D                                                           JAI 1400
      F1 = DAK1(J)                                                      JAI 1410
      F2 = 0.                                                           JAI 1420
      DO 20 I=1,M1D                                                     JAI 1430
        J = J - 1                                                       JAI 1440
        TEMP1 = F1                                                      JAI 1450
        F1 = TT*F1 - F2 + DAK1(J)                                       JAI 1460
        F2 = TEMP1                                                      JAI 1470
   20 CONTINUE                                                          JAI 1480
      DAI = -(T*F1-F2+DAK1(1))                                          JAI 1490
      RETURN                                                            JAI 1500
   30 CONTINUE                                                          JAI 1510
      T = (X+X-CON2)*CON3                                               JAI 1520
      TT = T + T                                                        JAI 1530
      J = N2                                                            JAI 1540
      F1 = AK2(J)                                                       JAI 1550
      F2 = 0.                                                           JAI 1560
      DO 40 I=1,M2                                                      JAI 1570
        J = J - 1                                                       JAI 1580
        TEMP1 = F1                                                      JAI 1590
        F1 = TT*F1 - F2 + AK2(J)                                        JAI 1600
```

```
      F2 = TEMP1                                                    JAI 161Ø
 4Ø CONTINUE                                                        JAI 162Ø
    RTRX = SQRT(RX)                                                 JAI 163Ø
    EC = EXP(-C)                                                    JAI 164Ø
    AI = EC*(T*F1-F2+AK2(1))/RTRX                                   JAI 165Ø
    J = N2D                                                         JAI 166Ø
    F1 = DAK2(J)                                                    JAI 167Ø
    F2 = Ø.                                                         JAI 168Ø
    DO 5Ø I=1,M2D                                                   JAI 169Ø
      J = J - 1                                                     JAI 17ØØ
      TEMP1 = F1                                                    JAI 171Ø
      F1 = TT*F1 - F2 + DAK2(J)                                     JAI 172Ø
      F2 = TEMP1                                                    JAI 173Ø
 5Ø CONTINUE                                                        JAI 174Ø
    DAI = -EC*(T*F1-F2+DAK2(1))*RTRX                                JAI 175Ø
    RETURN                                                          JAI 176Ø
 6Ø CONTINUE                                                        JAI 177Ø
    T = 1Ø./C - 1.                                                  JAI 178Ø
    TT = T + T                                                      JAI 179Ø
    J = N1                                                          JAI 18ØØ
    F1 = AK3(J)                                                     JAI 181Ø
    F2 = Ø.                                                         JAI 182Ø
    DO 7Ø I=1,M1                                                    JAI 183Ø
      J = J - 1                                                     JAI 184Ø
      TEMP1 = F1                                                    JAI 185Ø
      F1 = TT*F1 - F2 + AK3(J)                                      JAI 186Ø
      F2 = TEMP1                                                    JAI 187Ø
 7Ø CONTINUE                                                        JAI 188Ø
    RTRX = SQRT(RX)                                                 JAI 189Ø
    EC = EXP(-C)                                                    JAI 19ØØ
    AI = EC*(T*F1-F2+AK3(1))/RTRX                                   JAI 191Ø
    J = N1D                                                         JAI 192Ø
    F1 = DAK3(J)                                                    JAI 193Ø
    F2 = Ø.                                                         JAI 194Ø
    DO 8Ø I=1,M1D                                                   JAI 195Ø
      J = J - 1                                                     JAI 196Ø
      TEMP1 = F1                                                    JAI 197Ø
      F1 = TT*F1 - F2 + DAK3(J)                                     JAI 198Ø
      F2 = TEMP1                                                    JAI 199Ø
 8Ø CONTINUE                                                        JAI 2ØØØ
    DAI = -RTRX*EC*(T*F1-F2+DAK3(1))                                JAI 2Ø1Ø
    RETURN                                                          JAI 2Ø2Ø
 9Ø CONTINUE                                                        JAI 2Ø3Ø
    IF (C.GT.5.) GO TO 12Ø                                          JAI 2Ø4Ø
    T = .4*C - 1.                                                   JAI 2Ø5Ø
    TT = T + T                                                      JAI 2Ø6Ø
    J = N3                                                          JAI 2Ø7Ø
    F1 = AJP(J)                                                     JAI 2Ø8Ø
    E1 = AJN(J)                                                     JAI 2Ø9Ø
    F2 = Ø.                                                         JAI 21ØØ
    E2 = Ø.                                                         JAI 211Ø
    DO 1ØØ I=1,M3                                                   JAI 212Ø
      J = J - 1                                                     JAI 213Ø
      TEMP1 = F1                                                    JAI 214Ø
      TEMP2 = E1                                                    JAI 215Ø
      F1 = TT*F1 - F2 + AJP(J)                                      JAI 216Ø
      E1 = TT*E1 - E2 + AJN(J)                                      JAI 217Ø
      F2 = TEMP1                                                    JAI 218Ø
      E2 = TEMP2                                                    JAI 219Ø
1ØØ CONTINUE                                                        JAI 22ØØ
    AI = (T*E1-E2+AJN(1)) - X*(T*F1-F2+AJP(1))                      JAI 221Ø
    J = N3D                                                         JAI 222Ø
    F1 = DAJP(J)                                                    JAI 223Ø
    E1 = DAJN(J)                                                    JAI 224Ø
    F2 = Ø.                                                         JAI 225Ø
    E2 = Ø.                                                         JAI 226Ø
    DO 11Ø I=1,M3D                                                  JAI 227Ø
      J = J - 1                                                     JAI 228Ø
      TEMP1 = F1                                                    JAI 229Ø
      TEMP2 = E1                                                    JAI 23ØØ
      F1 = TT*F1 - F2 + DAJP(J)                                     JAI 231Ø
      E1 = TT*E1 - E2 + DAJN(J)                                     JAI 232Ø
      F2 = TEMP1                                                    JAI 233Ø
      E2 = TEMP2                                                    JAI 234Ø
11Ø CONTINUE                                                        JAI 235Ø
    DAI = X*X*(T*F1-F2+DAJP(1)) + (T*E1-E2+DAJN(1))                 JAI 236Ø
```

```
      RETURN                                                            JAI 237Ø
  12Ø CONTINUE                                                          JAI 238Ø
      T = 1Ø./C - 1.                                                    JAI 239Ø
      TT = T + T                                                        JAI 24ØØ
      J = N4                                                            JAI 241Ø
      F1 = A(J)                                                         JAI 242Ø
      E1 = B(J)                                                         JAI 243Ø
      F2 = Ø.                                                           JAI 244Ø
      E2 = Ø.                                                           JAI 245Ø
      DO 13Ø I=1,M4                                                     JAI 246Ø
        J = J - 1                                                       JAI 247Ø
        TEMP1 = F1                                                      JAI 248Ø
        TEMP2 = E1                                                      JAI 249Ø
        F1 = TT*F1 - F2 + A(J)                                          JAI 25ØØ
        E1 = TT*E1 - E2 + B(J)                                          JAI 251Ø
        F2 = TEMP1                                                      JAI 252Ø
        E2 = TEMP2                                                      JAI 253Ø
  13Ø CONTINUE                                                          JAI 254Ø
      TEMP1 = T*F1 - F2 + A(1)                                          JAI 255Ø
      TEMP2 = T*E1 - E2 + B(1)                                          JAI 256Ø
      RTRX = SQRT(RX)                                                   JAI 257Ø
      CV = C - FPI12                                                    JAI 258Ø
      CCV = COS(CV)                                                     JAI 259Ø
      SCV = SIN(CV)                                                     JAI 26ØØ
      AI = (TEMP1*CCV-TEMP2*SCV)/RTRX                                   JAI 261Ø
      J = N4D                                                           JAI 262Ø
      F1 = DA(J)                                                        JAI 263Ø
      E1 = DB(J)                                                        JAI 264Ø
      F2 = Ø.                                                           JAI 265Ø
      E2 = Ø.                                                           JAI 266Ø
      DO 14Ø I=1,M4D                                                    JAI 267Ø
        J = J - 1                                                       JAI 268Ø
        TEMP1 = F1                                                      JAI 269Ø
        TEMP2 = E1                                                      JAI 27ØØ
        F1 = TT*F1 - F2 + DA(J)                                         JAI 271Ø
        E1 = TT*E1 - E2 + DB(J)                                         JAI 272Ø
        F2 = TEMP1                                                      JAI 273Ø
        E2 = TEMP2                                                      JAI 274Ø
  14Ø CONTINUE                                                          JAI 275Ø
      TEMP1 = T*F1 - F2 + DA(1)                                         JAI 276Ø
      TEMP2 = T*E1 - E2 + DB(1)                                         JAI 277Ø
      E1 = CCV*CON5 + .5*SCV                                            JAI 278Ø
      E2 = SCV*CON5 - .5*CCV                                            JAI 279Ø
      DAI = (TEMP1*E1-TEMP2*E2)*RTRX                                    JAI 28ØØ
      RETURN                                                            JAI 281Ø
      END                                                               JAI 282Ø


      SUBROUTINE JBESS(ALPHA, N, X, Y)                                  JBE   1Ø
C                                                                       JBE   2Ø
C         A CDC 66ØØ SUBROUTINE                                         JBE   3Ø
C                                                                       JBE   4Ø
C     AUTHORS                                                           JBE   5Ø
C         D.E. AMOS, S.L. DANIEL, AND M.K. WESTON                       JBE   6Ø
C         SANDIA LABORATORIES                                           JBE   7Ø
C         JANUARY, 1975                                                 JBE   8Ø
C                                                                       JBE   9Ø
C     REFERENCES                                                        JBE  1ØØ
C         ABRAMOWITZ, M. AND STEGUN, I.A. HANDBOOK OF MATHEMATICAL      JBE  11Ø
C         FUNCTIONS. NBS APPLIED MATHEMATICS SERIES 55, U.S. GOVERNMENT JBE  12Ø
C         PRINTING OFFICE, WASHINGTON, D.C., CHAPTERS 9 AND 1Ø.         JBE  13Ø
C                                                                       JBE  14Ø
C         AMOS, D.E., DANIEL, S.L. AND WESTON, M.K. CDC 66ØØ            JBE  15Ø
C         SUBROUTINES IBESS AND JBESS FOR BESSEL FUNCTIONS              JBE  16Ø
C         I/SUB(NU)/(X) AND J/SUB(NU)/(X), X.GE.Ø, NU.GE.Ø.             JBE  17Ø
C         ACM TRANS. MATH. SOFTWARE, 1977.                              JBE  18Ø
C                                                                       JBE  19Ø
C         OLVER, F.W.J. TABLES FOR BESSEL FUNCTIONS OF MODERATE OR      JBE  2ØØ
C         LARGE ORDERS. NPL MATHEMATICAL TABLES, VOL 6. HER MAJESTY-S   JBE  21Ø
C         STATIONERY OFFICE, LONDON, 1962.                              JBE  22Ø
C                                                                       JBE  23Ø
C         OLVER, F.W.J. THE ASYMPTOTIC EXPANSION OF BESSEL FUNCTIONS OF JBE  24Ø
C         LARGE ORDER. PHIL. TRANS. A, 247, PP. 328-368, 1954.          JBE  25Ø
C                                                                       JBE  26Ø
C     ABSTRACT                                                          JBE  27Ø
C         JBESS COMPUTES AN N MEMBER SEQUENCE OF J BESSEL FUNCTIONS     JBE  28Ø
```

```
C         J/SUB(ALPHA+K-1)/(X), K=1,...,N FOR NON-NEGATIVE ALPHA AND X. JBE  29Ø
C         A COMBINATION OF THE POWER SERIES, THE ASYMPTOTIC EXPANSION   JBE  3ØØ
C         FOR X TO INFINITY AND THE UNIFORM ASYMPTOTIC EXPANSION FOR    JBE  31Ø
C         NU TO INFINITY ARE APPLIED OVER SUBDIVISIONS OF THE (NU,X)    JBE  32Ø
C         PLANE. FOR VALUES OF (NU,X) NOT COVERED BY ONE OF THESE       JBE  33Ø
C         FORMULAE, THE ORDER IS INCREMENTED OR DECREMENTED BY INTEGER  JBE  34Ø
C         VALUES INTO A REGION WHERE ONE OF THE FORMULAE APPLY. BACKWARDJBE  35Ø
C         RECURSION IS APPLIED TO REDUCE ORDERS BY INTEGER VALUES EXCEPTJBE  36Ø
C         WHERE THE ENTIRE SEQUENCE LIES IN THE OSCILLATORY REGION. IN  JBE  37Ø
C         THIS CASE FORWARD RECURSION IS STABLE AND VALUES FROM THE     JBE  38Ø
C         ASYMPTOTIC EXPANSION FOR X TO INFINITY START THE RECURSION    JBE  39Ø
C         WHEN IT IS EFFICIENT TO DO SO. LEADING TERMS OF THE SERIES ANDJBE  4ØØ
C         UNIFORM EXPANSION ARE TESTED FOR UNDERFLOW. IF A SEQUENCE IS  JBE  41Ø
C         REQUESTED AND THE LAST MEMBER WOULD UNDERFLOW, THE RESULT IS  JBE  42Ø
C         SET TO ZERO AND THE NEXT LOWER ORDER TRIED, ETC., UNTIL A     JBE  43Ø
C         MEMBER COMES ON SCALE OR ALL MEMBERS ARE SET TO ZERO. OVERFLOWJBE  44Ø
C         CANNOT OCCUR. JBESS CALLS SUBROUTINE JAIRY AND FUNCTION GAMLN.JBE  45Ø
C                                                                       JBE  46Ø
C     DESCRIPTION OF ARGUMENTS                                          JBE  47Ø
C                                                                       JBE  48Ø
C         INPUT                                                         JBE  49Ø
C           ALPHA  - ORDER OF FIRST MEMBER OF THE SEQUENCE, ALPHA.GE.Ø  JBE  5ØØ
C           N      - NUMBER OF MEMBERS IN THE SEQUENCE, N.GE.1          JBE  51Ø
C           X      - X.GE.Ø                                             JBE  52Ø
C                                                                       JBE  53Ø
C         OUTPUT                                                        JBE  54Ø
C           Y      - A VECTOR WHOSE FIRST N COMPONENTS CONTAIN          JBE  55Ø
C                    VALUES FOR J/SUB(ALPHA+K-1)/(X), K=1,...,N         JBE  56Ø
C                                                                       JBE  57Ø
C     ERROR CONDITIONS                                                  JBE  58Ø
C         IMPROPER INPUT ARGUMENTS - A FATAL ERROR                      JBE  59Ø
C         UNDERFLOW  - A NON-FATAL ERROR                                JBE  6ØØ
C                                                                       JBE  61Ø
      DOUBLE PRECISION DX, TRX, DTM, DFN                                JBE  62Ø
      DIMENSION Y(1)                                                    JBE  63Ø
      DIMENSION C(11,1Ø), ALFA(26,4), BETA(26,5)                        JBE  64Ø
      DIMENSION C1(11,8), C2(11,2), ALFA1(26,2), ALFA2(26,2)            JBE  65Ø
      DIMENSION BETA1(26,2), BETA2(26,2), BETA3(26,1)                   JBE  66Ø
      DIMENSION GAMA(26), TEMP(3), KMAX(5), AR(8), BR(1Ø), UPOL(1Ø)     JBE  67Ø
      DIMENSION FNULIM(2), PP(4)                                        JBE  68Ø
      DIMENSION CR(1Ø), DR(1Ø)                                          JBE  69Ø
      EQUIVALENCE (C(1,1),C1(1,1))                                      JBE  7ØØ
      EQUIVALENCE (C(1,9),C2(1,1))                                      JBE  71Ø
      EQUIVALENCE (ALFA(1,1),ALFA1(1,1))                                JBE  72Ø
      EQUIVALENCE (ALFA(1,3),ALFA2(1,1))                                JBE  73Ø
      EQUIVALENCE (BETA(1,1),BETA1(1,1))                                JBE  74Ø
      EQUIVALENCE (BETA(1,3),BETA2(1,1))                                JBE  75Ø
      EQUIVALENCE (BETA(1,5),BETA3(1,1))                                JBE  76Ø
      DATA ELIM1,ELIM2,TOL /   667.    ,    644.    ,    1.E-15        /JBE  77Ø
C                                                                       JBE  78Ø
      DATA PP              / 8.729Ø9153935547E+ØØ, 2.65693932265Ø3ØE-Ø1,JBE  79Ø
     1 1.24578576865586E-Ø1, 7.7Ø133747430388E-Ø4/                      JBE  8ØØ
C TOLS=LN(1.E-3)                                                        JBE  81Ø
      DATA TOLS            /-6.9Ø775527898214E+ØØ/                      JBE  82Ø
      DATA CON1,CON2,CON548/ 6.66666666666667E-Ø1, 3.33333333333333E-Ø1,JBE  83Ø
     1 1.Ø4166666666667E-Ø1/                                            JBE  84Ø
      DATA RTWO,PDF,RTTP,PIDT                    / 1.34839972492648E+ØØ,JBE  85Ø
     1 7.85398163397448E-Ø1, 7.9788456Ø8Ø2865E-Ø1, 1.57Ø79632679490E+ØØ/JBE  86Ø
      DATA FNULIM          /         1ØØ.       ,         6Ø.         /JBE  87Ø
C CE=-ALOG(TOL) , TCE=-Ø.75*ALOG(TOL)                                   JBE  88Ø
      DATA CE , TCE        / 3.4538776394910 7E+Ø1, 2.59Ø4Ø822961830E+Ø1/JBE  89Ø
      DATA INLIM           /         15Ø        /                      JBE  9ØØ
      DATA   AR            / 8.355Ø3472222222E-Ø2, 1.28226574556327E-Ø1,JBE  91Ø
     1 2.91849Ø26464140E-Ø1, 8.81627267443758E-Ø1, 3.3214Ø828186277E+ØØ,JBE  92Ø
     2 1.49957629868626E+Ø1, 7.8923Ø13Ø115865E+Ø1, 4.74451538868264E+Ø2/JBE  93Ø
      DATA   BR            /-1.45833333333333E-Ø1,-9.87413194444444E-Ø2,JBE  94Ø
     1-1.43312Ø53915895E-Ø1,-3.172272Ø2678414E-Ø1,-9.4242914795712ØE-Ø1,JBE  95Ø
     2-3.5112Ø3Ø4Ø82635E+ØØ,-1.572726362Ø368ØE+Ø1,-8.2281439Ø971859E+Ø1,JBE  96Ø
     3-4.9235537Ø523671E+Ø2,-3.31621856854797E+Ø3/                      JBE  97Ø
      DATA C1              /-2.Ø8333333333333E-Ø1, 1.25ØØØØØØØØØØØØØE-Ø1,JBE  98Ø
     1         9(Ø.)       , 3.342Ø1388888889E-Ø1,-4.Ø1Ø41666666667E-Ø1,JBE  99Ø
     2 7.Ø3125ØØØØØØØØØE-Ø2,         8(Ø.)       ,-1.Ø2581259645Ø62E+ØØ,JBE 1ØØØ
     3 1.84646267361111E+ØØ,-8.9121Ø9375ØØØØØE-Ø1, 7.32421875ØØØØØE-Ø2,JBE 1Ø1Ø
     4         7(Ø.)       , 4.66958442342625E+ØØ,-1.12Ø7ØØ26162230E+Ø1,JBE 1Ø2Ø
     5 8.78912353515625E+ØØ,-2.364Ø86914Ø625ØE+ØØ, 1.12152Ø996Ø9375E-Ø1,JBE 1Ø3Ø
     6         6(Ø.)       ,-2.8212Ø725582ØØ2E+Ø1, 8.463621767460Ø7E+Ø1,JBE 1Ø4Ø
```

```
7-9.1818241543240ØE+Ø1, 4.25349987453885E+Ø1,-7.36879435947963E+ØØ,JBE 1Ø5Ø
8 2.271Ø8ØØ17Ø8984E-Ø1,         5(Ø.)       , 2.1257Ø13ØØ39217E+Ø2,JBE 1Ø6Ø
9-7.65252468141182E+Ø2, 1.Ø5999Ø452528ØØE+Ø3,-6.99579627376133E+Ø2,JBE 1Ø7Ø
A 2.1819Ø511744212E+Ø2,-2.649143Ø4869516E+Ø1, 5.725Ø142Ø974731E-Ø1,JBE 1Ø8Ø
B         4(Ø.)       ,-1.91945766231841E+Ø3, 8.Ø6172218173731E+Ø3,JBE 1Ø9Ø
C-1.358655ØØØ64341E+Ø4, 1.16553933368645E+Ø4,-5.3Ø5646978613 4ØE+Ø3,JBE 11ØØ
D 1.2ØØ9Ø291321635E+Ø3,-1.Ø8Ø9Ø919788395E+Ø2, 1.7277275Ø258446E+ØØ,JBE 111Ø
E         3(Ø.)       , 2.Ø2Ø429133Ø9661E+Ø4,-9.698Ø5983886375E+Ø4,JBE 112Ø
F 1.92547Ø01232532E+Ø5,-2.Ø34ØØ177280416E+Ø5, 1.222ØØ464983Ø17E+Ø5,JBE 113Ø
G-4.11926549688976E+Ø4, 7.1Ø9514Ø248936E+Ø3,-4.939153Ø4773Ø88E+Ø2,JBE 114Ø
H 6.Ø74Ø42ØØ127348E+ØØ,         2(Ø.)       /                       JBE 115Ø
 DATA C2              /-2.4291918790Ø551E+Ø5, 1.31176361466298E+Ø6,JBE 116Ø
1-2.998Ø1591853811E+Ø6, 3.76327129765640E+Ø6,-2.81356322658653E+Ø6,JBE 117Ø
2 1.26836527332162E+Ø6,-3.31645172484564E+Ø5, 4.52187689813627E+Ø4,JBE 118Ø
3-2.49983Ø48181121E+Ø3, 2.438Ø5296995561E+Ø1,         1(Ø.)       ,JBE 119Ø
4 3.2844698530720 4E+Ø6,-1.970681911843 22E+Ø7, 5.Ø9526Ø24926646E+Ø7,JBE 12ØØ
5-7.41Ø51482115327E+Ø7, 6.6344512274729ØE+Ø7,-3.756717666Ø7634E+Ø7,JBE 121Ø
6 1.32887671664218E+Ø7,-2.7856181280 8645E+Ø6, 3.Ø81864Ø4612662E+Ø5,JBE 122Ø
7-1.3886Ø897537170E+Ø4, 1.1ØØ1714Ø269247E+Ø2/                       JBE 123Ø
 DATA ALFA1           /-4.44444444444444E-Ø3,-9.22Ø77922Ø77922E-Ø4,JBE 124Ø
1-8.84892884892885E-Ø5, 1.6592768783245ØE-Ø4, 2.46691372741793E-Ø4,JBE 125Ø
2 2.65995589346255E-Ø4, 2.6182429706 15Ø1E-Ø4, 2.4873Ø437344656E-Ø4,JBE 126Ø
3 2.32721Ø4ØØ83232E-Ø4, 2.16362485712365E-Ø4, 2.ØØ738858762752E-Ø4,JBE 127Ø
4 1.86267636637545E-Ø4, 1.73Ø6Ø775917876E-Ø4, 1.61Ø917Ø5929Ø16E-Ø4,JBE 128Ø
5 1.5Ø27477416Ø9Ø8E-Ø4, 1.4Ø5Ø349739127ØE-Ø4, 1.31668816545923E-Ø4,JBE 129Ø
6 1.23667445598253E-Ø4, 1.164Ø5271474738E-Ø4, 1.Ø9798298372713E-Ø4,JBE 13ØØ
7 1.Ø377241Ø422993E-Ø4, 9.82626Ø78369363E-Ø5, 9.3212Ø5172495Ø3E-Ø5,JBE 131Ø
8 8.8571Ø852478712E-Ø5, 8.429631Ø57157ØØE-Ø5, 8.Ø3497548407791E-Ø5,JBE 132Ø
9 6.93735541354589E-Ø4, 2.32241745182922E-Ø4,-1.41986273556691E-Ø5,JBE 133Ø
A-1.16444931672Ø49E-Ø4,-1.5Ø8Ø3558Ø53Ø49E-Ø4,-1.5512192491809 6E-Ø4,JBE 134Ø
B-1.468Ø97566466466E-Ø4,-1.338155Ø3867491E-Ø4,-1.19744975684254E-Ø4,JBE 135Ø
C-1.Ø6184319207974E-Ø4,-9.37699549891194E-Ø5,-8.26923Ø45588193E-Ø5,JBE 136Ø
D-7.29374348155221E-Ø5,-6.44Ø4235772 1Ø16E-Ø5,-5.69611566ØØ9369E-Ø5,JBE 137Ø
E-5.Ø4731Ø443Ø3562E-Ø5,-4.48134868ØØ8883E-Ø5,-3.98688727717599E-Ø5,JBE 138Ø
F-3.554ØØ532972Ø42E-Ø5,-3.174142566Ø9Ø22E-Ø5,-2.8399679390 4175E-Ø5,JBE 139Ø
G-2.5452272Ø634871E-Ø5,-2.28459297164725E-Ø5,-2.Ø5352753106481E-Ø5,JBE 14ØØ
H-1.84816217627666E-Ø5,-1.6651933ØØ21394E-Ø5/                       JBE 141Ø
 DATA ALFA2           /-3.54211971457744E-Ø4,-1.56161263945159E-Ø4,JBE 142Ø
1 3.Ø44655Ø3594936E-Ø5, 1.3Ø198655773243E-Ø4, 1.674711Ø6699712E-Ø4,JBE 143Ø
2 1.7Ø222587683593E-Ø4, 1.565Ø14276Ø8595E-Ø4, 1.3633917Ø977445E-Ø4,JBE 144Ø
3 1.14886692Ø29825E-Ø4, 9.45869Ø93Ø34688E-Ø5, 7.6449841925Ø898E-Ø5,JBE 145Ø
4 6.Ø757Ø334965197E-Ø5, 4.743942992 9Ø5Ø9E-Ø5, 3.6275751 2Ø05344E-Ø5,JBE 146Ø
5 2.69939714979225E-Ø5, 1.9321Ø938247939E-Ø5, 1.3ØØ56674793963E-Ø5,JBE 147Ø
6 7.8262Ø866744497E-Ø6, 3.59257485819352E-Ø6, 1.44Ø4ØØ49814252E-Ø7,JBE 148Ø
7-2.65396769697939E-Ø6,-4.91346867Ø98486E-Ø6,-6.72739296Ø91248E-Ø6,JBE 149Ø
8-8.17269379678658E-Ø6,-9.313Ø4715Ø93561E-Ø6,-1.Ø2Ø11418798Ø16E-Ø5,JBE 15ØØ
9 3.78194199201773E-Ø4, 2.Ø2471952761816E-Ø4,-6.379385Ø6318862E-Ø5,JBE 151Ø
A-2.3859823Ø6Ø3ØØ6E-Ø4,-3.1Ø916256Ø27362E-Ø4,-3.1368Ø115247576E-Ø4,JBE 152Ø
B-2.7895Ø273791323E-Ø4,-2.28564Ø82619141E-Ø4,-1.7524528Ø34Ø847E-Ø4,JBE 153Ø
C-1.25544Ø63Ø6Ø69ØE-Ø4,-8.2298287282Ø2Ø8E-Ø5,-4.6286Ø73Ø588116E-Ø5,JBE 154Ø
D-1.72334Ø2366962E-Ø5, 5.6Ø69Ø4823Ø46Ø2E-Ø6, 2.31395443148287E-Ø5,JBE 155Ø
E 3.62642745856794E-Ø5, 4.58ØØ6124490189E-Ø5, 5.245952949591 14E-Ø5,JBE 156Ø
F 5.683962Ø8545815E-Ø5, 5.9434982Ø3931Ø4E-Ø5, 6.Ø6478527578422E-Ø5,JBE 157Ø
G 6.Ø8Ø239Ø7788436E-Ø5, 6.Ø157789453946ØE-Ø5, 5.89199657344698E-Ø5,JBE 158Ø
H 5.72515823777593E-Ø5, 5.528Ø4375585853E-Ø5/                       JBE 159Ø
 DATA BETA1           / 1.79988721413553E-Ø2, 5.5996491 1Ø64388E-Ø3,JBE 16ØØ
1 2.885Ø14Ø2231133E-Ø3, 1.8ØØ966Ø6761Ø54E-Ø3, 1.2475311Ø589199E-Ø3,JBE 161Ø
2 9.22878876572938E-Ø4, 7.1443Ø421727287E-Ø4, 5.71787281789705E-Ø4,JBE 162Ø
3 4.69431ØØ76Ø6482E-Ø4, 3.93232835462917E-Ø4, 3.348188893182 98E-Ø4,JBE 163Ø
4 2.88952148495752E-Ø4, 2.52211615549573E-Ø4, 2.2228Ø58Ø798883E-Ø4,JBE 164Ø
5 1.97541838Ø33Ø63E-Ø4, 1.76836855Ø19718E-Ø4, 1.59316899661821E-Ø4,JBE 165Ø
6 1.4434793Ø197334E-Ø4, 1.31448Ø68119965E-Ø4, 1.2Ø2454449493Ø3E-Ø4,JBE 166Ø
7 1.1Ø4491445Ø4599E-Ø4, 1.Ø182877Ø74Ø567E-Ø4, 9.4199822 4Ø4238E-Ø5,JBE 167Ø
8 8.7413Ø545753834E-Ø5, 8.1346626216 28Ø1E-Ø5, 7.59ØØ2269646219E-Ø5,JBE 168Ø
9-1.49282953213429E-Ø3,-8.782Ø47Ø9546389E-Ø4,-5.Ø2916549572Ø35E-Ø4,JBE 169Ø
A-2.94822138512746E-Ø4,-1.7546399697Ø783E-Ø4,-1.Ø4ØØ855Ø46Ø816E-Ø4,JBE 17ØØ
B-5.96141953Ø46458E-Ø5,-3.12Ø38929Ø76Ø98E-Ø5,-1.26Ø8973598Ø23ØE-Ø5,JBE 171Ø
C-2.428926Ø857573ØE-Ø7, 8.Ø5996165414274E-Ø6, 1.365Ø7ØØ9262147E-Ø5,JBE 172Ø
D 1.73964125472926E-Ø5, 1.986729788421 34E-Ø5, 2.1446326379Ø823E-Ø5,JBE 173Ø
E 2.23954659232457E-Ø5, 2.28967783814713E-Ø5, 2.3Ø785389811178E-Ø5,JBE 174Ø
F 2.3Ø321976Ø8Ø9Ø9E-Ø5, 2.28236Ø7372Ø349E-Ø5, 2.25ØØ58811Ø5292E-Ø5,JBE 175Ø
G 2.2Ø981Ø15361991E-Ø5, 2.16418427448 1Ø4E-Ø5, 2.115Ø7649256221E-Ø5,JBE 176Ø
H 2.Ø6388749782171E-Ø5, 2.Ø1165241997Ø82E-Ø5/                       JBE 177Ø
 DATA BETA2           / 5.52213Ø76721293E-Ø4, 4.47932581552385E-Ø4,JBE 178Ø
1 2.7952Ø653992Ø21E-Ø4, 1.52468156198447E-Ø4, 6.932711Ø5657Ø44E-Ø5,JBE 179Ø
2 1.76258683Ø69991E-Ø5,-1.35744996343269E-Ø5,-3.1797241335Ø427E-Ø5,JBE 18ØØ
```

```
     3-4.18861861696693E-Ø5,-4.69ØØ4889379141E-Ø5,-4.87665447413787E-Ø5,JBE 181Ø
     4-4.87Ø1ØØ31186735E-Ø5,-4.7475562Ø89ØØ87E-Ø5,-4.55813Ø58138628E-Ø5,JBE 182Ø
     5-4.333Ø9644511266E-Ø5,-4.Ø923Ø19315775ØE-Ø5,-3.848226386Ø3221E-Ø5,JBE 183Ø
     6-3.6Ø857167535411E-Ø5,-3.377933Ø6123367E-Ø5,-3.1588856Ø77211ØE-Ø5,JBE 184Ø
     7-2.95269561750807E-Ø5,-2.75978914828336E-Ø5,-2.58ØØ6174666884E-Ø5,JBE 185Ø
     8-2.413Ø8356761280E-Ø5,-2.2582350951834 6E-Ø5,-2.11479656768913E-Ø5,JBE 186Ø
     9-4.74617796559960E-Ø4,-4.77864567147321E-Ø4,-3.2Ø39Ø228Ø67Ø38E-Ø4,JBE 187Ø
     A-1.611Ø5Ø16119962E-Ø4,-4.257781Ø1285435E-Ø5, 3.44571294294968E-Ø5,JBE 188Ø
     B 7.97Ø92684Ø75675E-Ø5, 1.Ø3138236708272E-Ø4, 1.1246677526220 4E-Ø4,JBE 189Ø
     C 1.131Ø36421Ø8481E-Ø4, 1.Ø8651634848774E-Ø4, 1.Ø1437951597662E-Ø4,JBE 19ØØ
     D 9.29298396593364E-Ø5, 8.4Ø293133Ø16Ø9ØE-Ø5, 7.52727991349134E-Ø5,JBE 191Ø
     E 6.69632521975731E-Ø5, 5.92564547323195E-Ø5, 5.22169308826976E-Ø5,JBE 192Ø
     F 4.58539485165361E-Ø5, 4.Ø1445513891487E-Ø5, 3.5Ø48173ØØ31328E-Ø5,JBE 193Ø
     G 3.Ø5157995Ø34347E-Ø5, 2.6495611995Ø516E-Ø5, 2.2936363369Ø998E-Ø5,JBE 194Ø
     H 1.97893Ø56664Ø22E-Ø5, 1.7ØØ91984636413E-Ø5/                     JBE 195Ø
      DATA BETA3           / 7.3646581Ø572578E-Ø4, 8.7279Ø8Ø5146194E-Ø4,JBE 196Ø
     1 6.22614862573135E-Ø4, 2.8599815419430 4E-Ø4, 3.84737672879366E-Ø6,JBE 197Ø
     2-1.879Ø6ØØ3636972E-Ø4,-2.976Ø3646594555E-Ø4,-3.45998126832656E-Ø4,JBE 198Ø
     3-3.5338247Ø916Ø38E-Ø4,-3.357156357750 49E-Ø4,-3.Ø43211247890 4ØE-Ø4,JBE 199Ø
     4-2.667227230 47613E-Ø4,-2.27654214122820E-Ø4,-1.89922611854562E-Ø4,JBE 2ØØØ
     5-1.55Ø58918599Ø94E-Ø4,-1.2377824Ø761874E-Ø4,-9.62926147717644E-Ø5,JBE 2Ø1Ø
     6-7.25178327714425E-Ø5,-5.22Ø7ØØ28895634E-Ø5,-3.5Ø34775Ø5119Ø1E-Ø5,JBE 2Ø2Ø
     7-2.Ø6489761Ø35552E-Ø5,-8.7Ø1Ø6Ø96849767E-Ø6, 1.1369868675100E-Ø6,JBE 2Ø3Ø
     8 9.16426474122779E-Ø6, 1.56477785428873E-Ø5, 2.Ø8223629482467E-Ø5/JBE 2Ø4Ø
      DATA GAMA            / 6.2996Ø524947437E-Ø1, 2.51984209978975E-Ø1,JBE 2Ø5Ø
     1 1.5479Ø3ØØ415656E-Ø1, 1.1Ø713Ø62416159E-Ø1, 8.573Ø9395527395E-Ø2,JBE 2Ø6Ø
     2 6.97161316958684E-Ø2, 5.86Ø85671893714E-Ø2, 5.Ø4698873536311E-Ø2,JBE 2Ø7Ø
     3 4.426ØØ58Ø689155E-Ø2, 3.9372Ø661543510E-Ø2, 3.54283195924455E-Ø2,JBE 2Ø8Ø
     4 3.218188575Ø2Ø98E-Ø2, 2.946462Ø791158E-Ø2, 2.71581677112934E-Ø2,JBE 2Ø9Ø
     5 2.51768272973862E-Ø2, 2.3457Ø7553Ø6Ø79E-Ø2, 2.195Ø839Ø1349Ø7E-Ø2,JBE 21ØØ
     6 2.Ø621Ø828235646E-Ø2, 1.9438824Ø897881E-Ø2, 1.8381Ø6338ØØ683E-Ø2,JBE 211Ø
     7 1.74293213231963E-Ø2, 1.65685837786612E-Ø2, 1.57865285987918E-Ø2,JBE 212Ø
     8 1.5Ø7295Ø1494Ø96E-Ø2, 1.4419325Ø839955E-Ø2, 1.38184Ø5735342E-Ø2/JBE 213Ø
C TEST INPUT ARGUMENTS                                                  JBE 214Ø
      KT = 1                                                            JBE 215Ø
      IF (N-1) 71Ø, 1Ø, 2Ø                                             JBE 216Ø
   1Ø KT = 2                                                            JBE 217Ø
   2Ø NN = N                                                            JBE 218Ø
      IF (X) 72Ø, 3Ø, 8Ø                                               JBE 219Ø
   3Ø IF (ALPHA) 7ØØ, 4Ø, 5Ø                                           JBE 22ØØ
   4Ø Y(1) = 1.                                                         JBE 221Ø
      IF (N.EQ.1) RETURN                                                JBE 222Ø
      I1 = 2                                                            JBE 223Ø
      GO TO 6Ø                                                          JBE 224Ø
   5Ø I1 = 1                                                            JBE 225Ø
   6Ø DO 7Ø I=I1,N                                                      JBE 226Ø
        Y(I) = Ø.                                                       JBE 227Ø
   7Ø CONTINUE                                                          JBE 228Ø
      RETURN                                                            JBE 229Ø
   8Ø CONTINUE                                                          JBE 23ØØ
      IF (ALPHA.LT.Ø.) GO TO 7ØØ                                        JBE 231Ø
      DFN = DBLE(FLOAT(N)) + DBLE(ALPHA) - 1.D+Ø                        JBE 232Ø
      FNU = DFN                                                         JBE 233Ø
      XO2 = X*.5                                                        JBE 234Ø
      SXO2 = XO2*XO2                                                    JBE 235Ø
C DECISION TREE FOR REGION WHERE SERIES, ASYMPTOTIC EXPANSION FOR X     JBE 236Ø
C TO INFINITY AND ASYMPTOTIC EXPANSION FOR NU TO INFINITY ARE           JBE 237Ø
C APPLIED.                                                              JBE 238Ø
      IF (SXO2.LE.(FNU+1.)) GO TO 9Ø                                    JBE 239Ø
      TA = AMAX1(2Ø.,FNU)                                               JBE 24ØØ
      IF (X.GT.TA) GO TO 12Ø                                            JBE 241Ø
      IF (X.GT.12.) GO TO 11Ø                                           JBE 242Ø
      XO2L = ALOG(XO2)                                                  JBE 243Ø
      NS = SXO2 - FNU                                                   JBE 244Ø
      GO TO 1ØØ                                                         JBE 245Ø
   9Ø FN = FNU                                                          JBE 246Ø
      FNP1 = FN + 1.                                                    JBE 247Ø
      XO2L = ALOG(XO2)                                                  JBE 248Ø
      IS = KT                                                           JBE 249Ø
      IF (X.LE.Ø.5) GO TO 33Ø                                           JBE 25ØØ
      NS = Ø                                                            JBE 251Ø
  1ØØ DFN = DFN + DBLE(FLOAT(NS))                                       JBE 252Ø
      FN = DFN                                                          JBE 253Ø
      FNP1 = FN + 1.                                                    JBE 254Ø
      IS = KT                                                           JBE 255Ø
      IF (N-1+NS.GT.Ø) IS = 3                                           JBE 256Ø
```

```
      GO TO 33Ø                                                          JBE 257Ø
   11Ø NS = AMAX1(36.-FNU,Ø.)                                            JBE 258Ø
      DFN = DFN + DBLE(FLOAT(NS))                                        JBE 259Ø
      FN = DFN                                                           JBE 26ØØ
      IS = KT                                                            JBE 261Ø
      IF (N-1+NS.GT.Ø) IS = 3                                            JBE 262Ø
      GO TO 13Ø                                                          JBE 263Ø
   12Ø CONTINUE                                                          JBE 264Ø
      RTX = SQRT(X)                                                      JBE 265Ø
      TAU = RTWO*RTX                                                     JBE 266Ø
      TA = TAU + FNULIM(KT)                                              JBE 267Ø
      IF (FNU.LE.TA) GO TO 48Ø                                           JBE 268Ø
      FN = FNU                                                           JBE 269Ø
      IS = KT                                                            JBE 27ØØ
C UNIFORM ASYMPTOTIC EXPANSION FOR NU TO INFINITY                        JBE 271Ø
   13Ø CONTINUE                                                          JBE 272Ø
      XX = X/FN                                                          JBE 273Ø
      W2 = 1. - XX*XX                                                    JBE 274Ø
      ABW2 = ABS(W2)                                                     JBE 275Ø
      RA = SQRT(ABW2)                                                    JBE 276Ø
      IF (ABW2.GT.Ø.2775) GO TO 22Ø                                      JBE 277Ø
C CASES NEAR X=FN, ABS(1.-(X/FN)**2).LE.Ø.2775                           JBE 278Ø
C COEFFICIENTS OF ASYMPTOTIC EXPANSION BY SERIES                         JBE 279Ø
C ZETA AND TRUNCATION FOR A(ZETA) AND B(ZETA) SERIES                     JBE 28ØØ
C KMAX IS TRUNCATION INDEX FOR A(ZETA) AND B(ZETA) SERIES=MAX(2,SA)      JBE 281Ø
      SA = Ø.                                                            JBE 282Ø
      IF (ABW2.EQ.Ø.) GO TO 14Ø                                          JBE 283Ø
      SA = TOLS/ALOG(ABW2)                                               JBE 284Ø
   14Ø SB = SA                                                           JBE 285Ø
      DO 15Ø I=1,5                                                       JBE 286Ø
        KMAX(I) = AMAX1(SA,2.)                                           JBE 287Ø
        SA = SA + SB                                                     JBE 288Ø
   15Ø CONTINUE                                                          JBE 289Ø
      KB = KMAX(5)                                                       JBE 29ØØ
      KLAST = KB - 1                                                     JBE 291Ø
      SA = GAMA(KB)                                                      JBE 292Ø
      DO 16Ø K=1,KLAST                                                   JBE 293Ø
        KB = KB - 1                                                      JBE 294Ø
        SA = SA*W2 + GAMA(KB)                                            JBE 295Ø
   16Ø CONTINUE                                                          JBE 296Ø
      Z = W2*SA                                                          JBE 297Ø
      AZ = ABS(Z)                                                        JBE 298Ø
      RTZ = SQRT(AZ)                                                     JBE 299Ø
      FN13 = FN**CON2                                                    JBE 3ØØØ
      RTARY = RTZ*FN13                                                   JBE 3Ø1Ø
      ARY = -RTARY*RTARY                                                 JBE 3Ø2Ø
      AZ32 = AZ*RTZ*CON1                                                 JBE 3Ø3Ø
      ACZ = FN*AZ32                                                      JBE 3Ø4Ø
      IF (Z.LE.Ø.) GO TO 17Ø                                             JBE 3Ø5Ø
C TEST FOR UNDERFLOW, 1.E-28Ø=EXP(-644.), ONE WORD LENGTH                JBE 3Ø6Ø
C UP FROM UNDERFLOW LIMIT OF CDC 66ØØ                                    JBE 3Ø7Ø
      IF (ACZ.GT.ELIM2) GO TO 38Ø                                        JBE 3Ø8Ø
      ARY = -ARY                                                         JBE 3Ø9Ø
   17Ø PHI = SQRT(SQRT(SA+SA+SA+SA))                                     JBE 31ØØ
C B(ZETA) FOR S=Ø                                                        JBE 311Ø
      KB = KMAX(5)                                                       JBE 312Ø
      KLAST = KB - 1                                                     JBE 313Ø
      SB = BETA(KB,1)                                                    JBE 314Ø
      DO 18Ø K=1,KLAST                                                   JBE 315Ø
        KB = KB - 1                                                      JBE 316Ø
        SB = SB*W2 + BETA(KB,1)                                          JBE 317Ø
   18Ø CONTINUE                                                          JBE 318Ø
      KSP1 = 1                                                           JBE 319Ø
      FN2 = FN*FN                                                        JBE 32ØØ
      RFN2 = 1./FN2                                                      JBE 321Ø
      RDEN = 1.                                                          JBE 322Ø
      ASUM = 1.                                                          JBE 323Ø
      RELB = TOL*ABS(SB)                                                 JBE 324Ø
      BSUM = SB                                                          JBE 325Ø
      DO 2ØØ KS=1,4                                                      JBE 326Ø
        KSP1 = KSP1 + 1                                                  JBE 327Ø
        RDEN = RDEN*RFN2                                                 JBE 328Ø
C A(ZETA) AND B(ZETA) FOR S=1,2,3,4                                      JBE 329Ø
        KB = KMAX(5-KS)                                                  JBE 33ØØ
        KLAST = KB - 1                                                   JBE 331Ø
        SA = ALFA(KB,KS)                                                 JBE 332Ø
```

```
      SB = BETA(KB,KSP1)                                              JBE 333Ø
      DO 19Ø K=1,KLAST                                                JBE 334Ø
        KB = KB - 1                                                   JBE 335Ø
        SA = SA*W2 + ALFA(KB,KS)                                      JBE 336Ø
        SB = SB*W2 + BETA(KB,KSP1)                                    JBE 337Ø
19Ø   CONTINUE                                                        JBE 338Ø
      TA = SA*RDEN                                                    JBE 339Ø
      TB = SB*RDEN                                                    JBE 34ØØ
      ASUM = ASUM + TA                                                JBE 341Ø
      BSUM = BSUM + TB                                                JBE 342Ø
      IF (ABS(TA).LE.TOL .AND. ABS(TB).LE.RELB) GO TO 21Ø             JBE 343Ø
2ØØ CONTINUE                                                          JBE 344Ø
21Ø CONTINUE                                                          JBE 345Ø
    BSUM = BSUM/(FN*FN13)                                             JBE 346Ø
    GO TO 3ØØ                                                         JBE 347Ø
22Ø CONTINUE                                                          JBE 348Ø
    TAU = 1./RA                                                       JBE 349Ø
    T2 = 1./W2                                                        JBE 35ØØ
    IF (W2.GE.Ø.) GO TO 23Ø                                           JBE 351Ø
C CASES FOR (X/FN).GT.SQRT(1.2775)                                    JBE 352Ø
    AZ32 = ABS(RA-ATAN(RA))                                           JBE 353Ø
    ACZ = AZ32*FN                                                     JBE 354Ø
    CZ = -ACZ                                                         JBE 355Ø
    Z32 = 1.5*AZ32                                                    JBE 356Ø
    RTZ = Z32**CON2                                                   JBE 357Ø
    FN13 = FN**CON2                                                   JBE 358Ø
    RTARY = RTZ*FN13                                                  JBE 359Ø
    ARY = -RTARY*RTARY                                                JBE 36ØØ
    GO TO 24Ø                                                         JBE 361Ø
23Ø CONTINUE                                                          JBE 362Ø
C CASES FOR (X/FN).LT.SQRT(Ø.7225)                                    JBE 363Ø
    AZ32 = ABS(ALOG((1.+RA)/XX)-RA)                                   JBE 364Ø
C TEST FOR UNDERFLOW, 1.E-28Ø = EXP(-644.), ONE WORD LENGTH           JBE 365Ø
C UP FROM UNDERFLOW LIMIT OF CDC 66ØØ                                 JBE 366Ø
    ACZ = AZ32*FN                                                     JBE 367Ø
    CZ = ACZ                                                          JBE 368Ø
    IF (ACZ.GT.ELIM2) GO TO 38Ø                                       JBE 369Ø
    Z32 = 1.5*AZ32                                                    JBE 37ØØ
    RTZ = Z32**CON2                                                   JBE 371Ø
    FN13 = FN**CON2                                                   JBE 372Ø
    RTARY = RTZ*FN13                                                  JBE 373Ø
    ARY = RTARY*RTARY                                                 JBE 374Ø
24Ø CONTINUE                                                          JBE 375Ø
    PHI = SQRT((RTZ+RTZ)*TAU)                                         JBE 376Ø
    TB = 1.                                                           JBE 377Ø
    ASUM = 1.                                                         JBE 378Ø
    TFN = TAU/FN                                                      JBE 379Ø
    UPOL(1) = 1.                                                      JBE 38ØØ
    UPOL(2) = (C(1,1)*T2+C(2,1))*TFN                                  JBE 381Ø
    RCZ = CON1/CZ                                                     JBE 382Ø
    CRZ32 = CON548*RCZ                                                JBE 383Ø
    BSUM = UPOL(2) + CRZ32                                            JBE 384Ø
    RELB = TOL*ABS(BSUM)                                              JBE 385Ø
    AP = TFN                                                          JBE 386Ø
    KS = Ø                                                            JBE 387Ø
    KP1 = 2                                                           JBE 388Ø
    RZDEN = RCZ                                                       JBE 389Ø
    DO 28Ø LR=2,8,2                                                   JBE 39ØØ
C COMPUTE TWO U POLYNOMIALS FOR NEXT A(ZETA) AND B(ZETA)              JBE 391Ø
      LRP1 = LR + 1                                                   JBE 392Ø
      DO 26Ø K=LR,LRP1                                                JBE 393Ø
        KS = KS + 1                                                   JBE 394Ø
        KP1 = KP1 + 1                                                 JBE 395Ø
        S1 = C(1,K)                                                   JBE 396Ø
        DO 25Ø J=2,KP1                                                JBE 397Ø
          S1 = S1*T2 + C(J,K)                                         JBE 398Ø
25Ø     CONTINUE                                                      JBE 399Ø
        AP = AP*TFN                                                   JBE 4ØØØ
        UPOL(KP1) = AP*S1                                             JBE 4Ø1Ø
        CR(KS) = BR(KS)*RZDEN                                         JBE 4Ø2Ø
        RZDEN = RZDEN*RCZ                                             JBE 4Ø3Ø
        DR(KS) = AR(KS)*RZDEN                                         JBE 4Ø4Ø
26Ø   CONTINUE                                                        JBE 4Ø5Ø
      SUMA = UPOL(LRP1)                                               JBE 4Ø6Ø
      SUMB = UPOL(LR+2) + UPOL(LRP1)*CRZ32                            JBE 4Ø7Ø
      JU = LRP1                                                       JBE 4Ø8Ø
```

```
            DO 27Ø JR=1,LR                                                  JBE 4Ø9Ø
              JU = JU - 1                                                   JBE 41ØØ
              SUMA = SUMA + CR(JR)*UPOL(JU)                                 JBE 411Ø
              SUMB = SUMB + DR(JR)*UPOL(JU)                                 JBE 412Ø
     27Ø    CONTINUE                                                        JBE 413Ø
            TB = -TB                                                        JBE 414Ø
            IF (W2.GT.Ø.) TB = ABS(TB)                                      JBE 415Ø
            ASUM = ASUM + SUMA*TB                                           JBE 416Ø
            BSUM = BSUM + SUMB*TB                                           JBE 417Ø
            IF (ABS(SUMA).LE.TOL .AND. ABS(SUMB).LE.RELB) GO TO 29Ø         JBE 418Ø
     28Ø CONTINUE                                                           JBE 419Ø
     29Ø TB = RTARY                                                         JBE 42ØØ
         IF (W2.GT.Ø.) TB = -TB                                             JBE 421Ø
         BSUM = BSUM/TB                                                     JBE 422Ø
     3ØØ CONTINUE                                                           JBE 423Ø
         CALL JAIRY(ARY, RTARY, ACZ, AI, DAI)                               JBE 424Ø
         TEMP(IS) = PHI*(AI*ASUM+DAI*BSUM)/FN13                             JBE 425Ø
         GO TO (32Ø, 45Ø, 61Ø), IS                                          JBE 426Ø
     31Ø TEMP(1) = TEMP(3)                                                  JBE 427Ø
         KT = 1                                                             JBE 428Ø
     32Ø IS = 2                                                             JBE 429Ø
         DFN = DFN - 1.D+Ø                                                  JBE 43ØØ
         FN = DFN                                                           JBE 431Ø
         GO TO 13Ø                                                          JBE 432Ø
C SERIES FOR (X/2)**2.LE.NU+1                                               JBE 433Ø
     33Ø CONTINUE                                                           JBE 434Ø
         GLN = GAMLN(FNP1)                                                  JBE 435Ø
         ARG = FN*XO2L - GLN                                                JBE 436Ø
         IF (ARG.LT.-ELIM1) GO TO 4ØØ                                       JBE 437Ø
         EARG = EXP(ARG)                                                    JBE 438Ø
     34Ø CONTINUE                                                           JBE 439Ø
         S = 1.                                                             JBE 44ØØ
         AK = 3.                                                            JBE 441Ø
         T2 = 1.                                                            JBE 442Ø
         T = 1.                                                             JBE 443Ø
         S1 = FN                                                            JBE 444Ø
         DO 35Ø K=1,17                                                      JBE 445Ø
           S2 = T2 + S1                                                     JBE 446Ø
           T = -T*SXO2/S2                                                   JBE 447Ø
           S = S + T                                                        JBE 448Ø
           IF (ABS(T).LT.TOL) GO TO 36Ø                                     JBE 449Ø
           T2 = T2 + AK                                                     JBE 45ØØ
           AK = AK + 2.                                                     JBE 451Ø
           S1 = S1 + FN                                                     JBE 452Ø
     35Ø CONTINUE                                                           JBE 453Ø
     36Ø CONTINUE                                                           JBE 454Ø
         TEMP(IS) = S*EARG                                                  JBE 455Ø
         GO TO (37Ø, 45Ø, 6ØØ), IS                                          JBE 456Ø
     37Ø EARG = EARG*FN/XO2                                                 JBE 457Ø
         DFN = DFN - 1.D+Ø                                                  JBE 458Ø
         FN = DFN                                                           JBE 459Ø
         IS = 2                                                             JBE 46ØØ
         GO TO 34Ø                                                          JBE 461Ø
C SET UNDERFLOW VALUE AND UPDATE PARAMETERS                                 JBE 462Ø
     38Ø Y(NN) = Ø.                                                         JBE 463Ø
         NN = NN - 1                                                        JBE 464Ø
         DFN = DFN - 1.D+Ø                                                  JBE 465Ø
         FN = DFN                                                           JBE 466Ø
         IF (NN-1) 44Ø, 39Ø, 13Ø                                            JBE 467Ø
     39Ø KT = 2                                                             JBE 468Ø
         IS = 2                                                             JBE 469Ø
         GO TO 13Ø                                                          JBE 47ØØ
     4ØØ Y(NN) = Ø.                                                         JBE 471Ø
         NN = NN - 1                                                        JBE 472Ø
         FNP1 = FN                                                          JBE 473Ø
         DFN = DFN - 1.D+Ø                                                  JBE 474Ø
         FN = DFN                                                           JBE 475Ø
         IF (NN-1) 44Ø, 41Ø, 42Ø                                            JBE 476Ø
     41Ø KT = 2                                                             JBE 477Ø
         IS = 2                                                             JBE 478Ø
     42Ø IF (SXO2.LE.FNP1) GO TO 43Ø                                        JBE 479Ø
         GO TO 13Ø                                                          JBE 48ØØ
     43Ø ARG = ARG - XO2L + ALOG(FNP1)                                      JBE 481Ø
         IF (ARG.LT.-ELIM1) GO TO 4ØØ                                       JBE 482Ø
         GO TO 33Ø                                                          JBE 483Ø
     44Ø NZ = N - NN                                                        JBE 484Ø
```

```
      PRINT 99996, NZ, ALPHA, N, X                                    JBE 485Ø
      RETURN                                                          JBE 486Ø
C BACKWARD RECURSION SECTION                                          JBE 487Ø
  45Ø CONTINUE                                                        JBE 488Ø
      NZ = N - NN                                                     JBE 489Ø
      IF (NZ.NE.Ø) PRINT 99996, NZ, ALPHA, N, X                       JBE 49ØØ
      IF (KT.EQ.2) GO TO 47Ø                                          JBE 491Ø
C BACKWARD RECUR FROM INDEX ALPHA+NN-1 TO ALPHA                       JBE 492Ø
      Y(NN) = TEMP(1)                                                 JBE 493Ø
      Y(NN-1) = TEMP(2)                                               JBE 494Ø
      IF (NN.EQ.2) RETURN                                             JBE 495Ø
      DX = X                                                          JBE 496Ø
      TRX = 2.D+Ø/DX                                                  JBE 497Ø
      DTM = DFN*TRX                                                   JBE 498Ø
      TM = DTM                                                        JBE 499Ø
      K = NN + 1                                                      JBE 5ØØØ
      DO 46Ø I=3,NN                                                   JBE 5Ø1Ø
        K = K - 1                                                     JBE 5Ø2Ø
        Y(K-2) = TM*Y(K-1) - Y(K)                                     JBE 5Ø3Ø
        DTM = DTM - TRX                                               JBE 5Ø4Ø
        TM = DTM                                                      JBE 5Ø5Ø
  46Ø CONTINUE                                                        JBE 5Ø6Ø
      RETURN                                                          JBE 5Ø7Ø
  47Ø Y(1) = TEMP(2)                                                  JBE 5Ø8Ø
      RETURN                                                          JBE 5Ø9Ø
C ASYMPTOTIC EXPANSION FOR X TO INFINITY WITH FORWARD RECURSION IN     JBE 51ØØ
C OSCILLATORY REGION X.GT.MAX(2Ø, NU), PROVIDED THE LAST MEMBER       JBE 511Ø
C OF THE SEQUENCE IS ALSO IN THE REGION.                              JBE 512Ø
  48Ø CONTINUE                                                        JBE 513Ø
      IN = ALPHA - TAU + 2.                                           JBE 514Ø
      IF (IN.LE.Ø) GO TO 49Ø                                          JBE 515Ø
      INP1 = IN + 1                                                   JBE 516Ø
      DALPHA = ALPHA - FLOAT(INP1)                                    JBE 517Ø
      KT = 1                                                          JBE 518Ø
      GO TO 5ØØ                                                       JBE 519Ø
  49Ø DALPHA = ALPHA                                                  JBE 52ØØ
      IN = Ø                                                          JBE 521Ø
  5ØØ IS = KT                                                         JBE 522Ø
      ARG = X - PIDT*DALPHA - PDF                                     JBE 523Ø
      SA = SIN(ARG)                                                   JBE 524Ø
      SB = COS(ARG)                                                   JBE 525Ø
      RA = RTTP/RTX                                                   JBE 526Ø
      ETX = 8.*X                                                      JBE 527Ø
  51Ø DX = DALPHA                                                     JBE 528Ø
      DX = DX + DX                                                    JBE 529Ø
      DTM = DX*DX                                                     JBE 53ØØ
      T2 = DTM - 1.D+Ø                                                JBE 531Ø
      T2 = T2/ETX                                                     JBE 532Ø
      S2 = T2                                                         JBE 533Ø
      RELB = TOL*ABS(T2)                                              JBE 534Ø
      T1 = ETX                                                        JBE 535Ø
      S1 = 1.                                                         JBE 536Ø
      FN = 1.                                                         JBE 537Ø
      AK = 8.                                                         JBE 538Ø
      DO 52Ø K=1,13                                                   JBE 539Ø
        T1 = T1 + ETX                                                 JBE 54ØØ
        FN = FN + AK                                                  JBE 541Ø
        DX = FN                                                       JBE 542Ø
        TRX = DTM - DX                                                JBE 543Ø
        AP = TRX                                                      JBE 544Ø
        T2 = -T2*AP/T1                                                JBE 545Ø
        S1 = S1 + T2                                                  JBE 546Ø
        T1 = T1 + ETX                                                 JBE 547Ø
        AK = AK + 8.                                                  JBE 548Ø
        FN = FN + AK                                                  JBE 549Ø
        DX = FN                                                       JBE 55ØØ
        TRX = DTM - DX                                                JBE 551Ø
        AP = TRX                                                      JBE 552Ø
        T2 = T2*AP/T1                                                 JBE 553Ø
        S2 = S2 + T2                                                  JBE 554Ø
        IF (ABS(T2).LE.RELB) GO TO 53Ø                                JBE 555Ø
        AK = AK + 8.                                                  JBE 556Ø
  52Ø CONTINUE                                                        JBE 557Ø
  53Ø TEMP(IS) = RA*(S1*SB-S2*SA)                                     JBE 558Ø
      GO TO (54Ø, 55Ø), IS                                            JBE 559Ø
  54Ø DALPHA = DALPHA + 1.                                            JBE 56ØØ
```

```
      IS = 2                                                          JBE 561Ø
      TB = SA                                                         JBE 562Ø
      SA = -SB                                                        JBE 563Ø
      SB = TB                                                         JBE 564Ø
      GO TO 51Ø                                                       JBE 565Ø
C FORWARD RECURSION SECTION                                           JBE 566Ø
 55Ø IF (KT.EQ.2) GO TO 47Ø                                           JBE 567Ø
      S1 = TEMP(1)                                                    JBE 568Ø
      S2 = TEMP(2)                                                    JBE 569Ø
      TX = 2./X                                                       JBE 57ØØ
      TM = DALPHA*TX                                                  JBE 571Ø
      IF (IN.EQ.Ø) GO TO 57Ø                                          JBE 572Ø
C FORWARD RECUR TO INDEX ALPHA                                        JBE 573Ø
      DO 56Ø I=1,IN                                                   JBE 574Ø
        S = S2                                                        JBE 575Ø
        S2 = TM*S2 - S1                                               JBE 576Ø
        TM = TM + TX                                                  JBE 577Ø
        S1 = S                                                        JBE 578Ø
 56Ø CONTINUE                                                         JBE 579Ø
      IF (NN.EQ.1) GO TO 59Ø                                          JBE 58ØØ
      S = S2                                                          JBE 581Ø
      S2 = TM*S2 - S1                                                 JBE 582Ø
      TM = TM + TX                                                    JBE 583Ø
      S1 = S                                                          JBE 584Ø
 57Ø CONTINUE                                                         JBE 585Ø
C FORWARD RECUR FROM INDEX ALPHA TO ALPHA+N-1                         JBE 586Ø
      Y(1) = S1                                                       JBE 587Ø
      Y(2) = S2                                                       JBE 588Ø
      IF (NN.EQ.2) RETURN                                             JBE 589Ø
      DO 58Ø I=3,NN                                                   JBE 59ØØ
        Y(I) = TM*Y(I-1) - Y(I-2)                                     JBE 591Ø
        TM = TM + TX                                                  JBE 592Ø
 58Ø CONTINUE                                                         JBE 593Ø
      RETURN                                                          JBE 594Ø
 59Ø Y(1) = S2                                                        JBE 595Ø
      RETURN                                                          JBE 596Ø
C BACKWARD RECURSION WITH NORMALIZATION BY                            JBE 597Ø
C ASYMPTOTIC EXPANSION FOR NU TO INFINITY OR POWER SERIES.            JBE 598Ø
 6ØØ CONTINUE                                                         JBE 599Ø
C COMPUTATION OF LAST ORDER FOR SERIES NORMALIZATION                  JBE 6ØØØ
      KM = AMAX1(3.-FN,Ø.)                                            JBE 6Ø1Ø
      TFN = FN + FLOAT(KM)                                            JBE 6Ø2Ø
      TA = (GLN+TFN-Ø.9189385332-Ø.Ø833333333/TFN)/(TFN+Ø.5)          JBE 6Ø3Ø
      TA = XO2L - TA                                                  JBE 6Ø4Ø
      TB = -(1.-1.5/TFN)/TFN                                          JBE 6Ø5Ø
      IN = CE/(-TA+SQRT(TA*TA-CE*TB)) + 1.5                           JBE 6Ø6Ø
      IN = IN + KM                                                    JBE 6Ø7Ø
      GO TO 65Ø                                                       JBE 6Ø8Ø
 61Ø CONTINUE                                                         JBE 6Ø9Ø
C COMPUTATION OF LAST ORDER FOR ASYMPTOTIC EXPANSION NORMALIZATION     JBE 61ØØ
      GLN = AZ32 + RA                                                 JBE 611Ø
      IF (ARY.GT.3Ø.) GO TO 63Ø                                       JBE 612Ø
      RDEN = (PP(4)*ARY+PP(3))*ARY + 1.                               JBE 613Ø
      RZDEN = PP(1) + PP(2)*ARY                                       JBE 614Ø
      TA = RZDEN/RDEN                                                 JBE 615Ø
      IF (W2.LT.Ø.1Ø) GO TO 62Ø                                       JBE 616Ø
      TB = GLN/RTARY                                                  JBE 617Ø
      GO TO 64Ø                                                       JBE 618Ø
 62Ø TB = (1.259921Ø49+Ø.167989473Ø*W2)/FN13                          JBE 619Ø
      GO TO 64Ø                                                       JBE 62ØØ
 63Ø CONTINUE                                                         JBE 621Ø
      TA = CON1*TCE/ACZ                                               JBE 622Ø
      TA = ((Ø.Ø493827160*TA-Ø.1111111111)*TA+Ø.6666656667)*TA*ARY    JBE 623Ø
      IF (W2.LT.Ø.1Ø) GO TO 62Ø                                       JBE 624Ø
      TB = GLN/RTARY                                                  JBE 625Ø
 64Ø IN = TA/TB + 1.5                                                 JBE 626Ø
      IF (IN.GT.INLIM) GO TO 31Ø                                      JBE 627Ø
 65Ø DX = FLOAT(IN)                                                   JBE 628Ø
      DTM = DFN + DX                                                  JBE 629Ø
      DX = X                                                          JBE 63ØØ
      TRX = 2.D+Ø/DX                                                  JBE 631Ø
      DTM = DTM*TRX                                                   JBE 632Ø
      TM = DTM                                                        JBE 633Ø
      TA = Ø.                                                         JBE 634Ø
      TB = TOL                                                        JBE 635Ø
      KK = 1                                                          JBE 636Ø
```

```
  66Ø CONTINUE                                                          JBE 637Ø
C BACKWARD RECUR UNINDEXED                                              JBE 638Ø
      DO 67Ø I=1,IN                                                     JBE 639Ø
        TB = TM*TB - TA                                                 JBE 64ØØ
        DTM = DTM - TRX                                                 JBE 641Ø
        TM = DTM                                                        JBE 642Ø
  67Ø CONTINUE                                                          JBE 643Ø
C NORMALIZATION                                                         JBE 644Ø
      IF (KK.NE.1) GO TO 68Ø                                            JBE 645Ø
      TA = (TA/TB)*TEMP(3)                                              JBE 646Ø
      TB = TEMP(3)                                                      JBE 647Ø
      KK = 2                                                            JBE 648Ø
      IN = NS                                                           JBE 649Ø
      IF (NS.NE.Ø) GO TO 66Ø                                            JBE 65ØØ
  68Ø Y(NN) = TB                                                        JBE 651Ø
      NZ = N - NN                                                       JBE 652Ø
      IF (NZ.NE.Ø) PRINT 99996, NZ, ALPHA, N, X                         JBE 653Ø
      IF (NN.EQ.1) RETURN                                               JBE 654Ø
      TB = TM*TB - TA                                                   JBE 655Ø
      DTM = DTM - TRX                                                   JBE 656Ø
      TM = DTM                                                          JBE 657Ø
      K = NN - 1                                                        JBE 658Ø
      Y(K) = TB                                                         JBE 659Ø
      IF (NN.EQ.2) RETURN                                               JBE 66ØØ
      KM = K - 1                                                        JBE 661Ø
C BACKWARD RECUR INDEXED                                                JBE 662Ø
      DO 69Ø I=1,KM                                                     JBE 663Ø
        Y(K-1) = TM*Y(K) - Y(K+1)                                       JBE 664Ø
        DTM = DTM - TRX                                                 JBE 665Ø
        TM = DTM                                                        JBE 666Ø
        K = K - 1                                                       JBE 667Ø
  69Ø CONTINUE                                                          JBE 668Ø
      RETURN                                                            JBE 669Ø
  7ØØ PRINT 99999, ALPHA, N, X                                          JBE 67ØØ
      STOP                                                              JBE 671Ø
  71Ø PRINT 99998, ALPHA, N, X                                          JBE 672Ø
      STOP                                                              JBE 673Ø
  72Ø PRINT 99997, ALPHA, N, X                                          JBE 674Ø
      STOP                                                              JBE 675Ø
99999 FORMAT (51HØJBESS CALLED WITH THE ORDER, ALPHA, LESS THAN ZERO     JBE 676Ø
     * /7H ALPHA=,E25.14,3H N=,I6,3H X=,E25.14)                         JBE 677Ø
99998 FORMAT (34HØJBESS CALLED WITH N LESS THAN ONE/7H ALPHA=,           JBE 678Ø
     * E25.14,3H N=,I6,3H X=,E25.14)                                    JBE 679Ø
99997 FORMAT (35HØJBESS CALLED WITH X LESS THAN ZERO/7H ALPHA=,          JBE 68ØØ
     * E25.14,3H N=,I6,3H X=,E25.14)                                    JBE 681Ø
99996 FORMAT (25HØUNDERFLOW IN JBESS, LAST,I6,2ØH VALUE(S) OF Y ARRAY,  JBE 682Ø
     * 17H WERE SET TO ZERO/7H ALPHA=,E25.14,3H N=,I6,3H X=,E25.14)     JBE 683Ø
      END                                                               JBE 684Ø
```

ERRATUM: ALGORITHM 511

CDC 6600 Subroutines IBESS and JBESS for Bessel Functions $I_\nu(x)$ and $J_\nu(x)$, $x \geq 0$, $\nu \geq 0$ [S18]

[D.E. Amos, S.L. Daniel, and M.K. Weston, *ACM Trans. Math. Software 3,* 1 (March 1977), 93–95]

Donald E. Amos [Recd 28 July 1977]
Numerical Mathematics Division 5122, Sandia Laboratories, Albuquerque, NM 87115

The following changes should be made in this algorithm:

(1) Add (immediately after line JBE 6390)[1]

```
      S = TB          JBE 6391
```

[1] To make this change, see "Collected Algorithms from ACM," pages 511-P1 through 511-P21.

(2) Add (immediately after line JBE 6400)[1]

```
TA = S          JBE 6401
```

(3) Replace the date

```
1977
```

by the date

```
March, 1977
```

in the lines IBE 180 and JBE 180,[2] and the date

```
1976
```

by the date

```
March, 1977
```

in the line GAM 180.[1]

[2] To make these changes, see the issue of *ACM Trans. Math. Software* referenced above, pages 94 and 95.

ALGORITHM 512
A Normalized Algorithm for the Solution of Positive Definite Symmetric Quindiagonal Systems of Linear Equations [F4]

A. BENSON and D. J. EVANS
Loughborough University of Technology, England

Key Words and Phrases: linear equations, normalized solution, periodic quindiagonal, symmetric positive definite
CR Categories: 5.14, 5.17
Language: Fortran

DESCRIPTION

In the finite difference solution of elliptic partial differential equations using block iterative methods, there occurs within the iteration scheme a need to solve many subsets of the resulting set of linear equations for each block and for each iteration. The form of the coefficient matrix B of such subsets of equations depends on the particular differential equation and the region of integration and on the way in which the grid points in the finite difference scheme are ordered. In this algorithm we are concerned with the case when B is symmetric, positive definite, and of the form

$$\begin{pmatrix} c_1 & b_1 & a_1 & & & & & a_{N-1} & b_N \\ b_1 & c_2 & b_2 & a_2 & & & & & a_N \\ a_1 & b_2 & c_3 & b_3 & a_3 & & & & \\ & a_2 & & & & & & & \\ & & \ddots & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & \ddots & & \\ & & & & \ddots & \ddots & \ddots & \ddots & \\ & & & a_{N-4} & b_{N-3} & c_{N-2} & b_{N-2} & a_{N-2} & \\ a_{N-1} & & & & a_{N-3} & b_{N-2} & c_{N-1} & b_{N-1} & \\ b_N & a_N & & & & a_{N-2} & b_{N-1} & c_N & \end{pmatrix} \tag{1}$$

Such systems arise, for example, from a two-line grouping of the mesh points for a self-adjoint partial differential equation in a rectangular region with periodic boundary conditions in one or both coordinate directions. Similar systems also arise in the solution of the biharmonic equation in a rectangular region subject to periodicity conditions [4]. For nonsymmetric matrices such as might occur if the differential equation were not self-adjoint or for circular regions [1], an algorithm has been given [2]. For the symmetric, positive definite case, however, it is worth-

Received 13 December 1974 and 4 November 1975.

Authors' address: Department of Computer Studies, Loughborough University of Technology, Leicestershire, England.

while to consider a normalized form of the algorithm following Cuthill and Varga [3]. This approach makes use of the result:

If A is a real $N \times N$, symmetric positive definite matrix, then there exists a unique, positive, diagonal matrix D and a unique, real, upper triangular matrix T, with unit diagonal elements such that

$$A = DT'TD \tag{2}$$

where T' denotes the transpose of T.

Assume, then, that in the system of equations

$$B\phi = \mathbf{s} \tag{3}$$

the matrix B is symmetric, positive definite, and of the form given by (1). Then B has the unique factorization

$$B = DT'TD$$

where

$$D = \text{diag}\,\{d_1, d_2, d_3, \ldots, d_N\} \tag{4}$$

and T is given by

$$T = \begin{pmatrix}
1 & e_1 & f_1 & & & & & g_1 & & h_1 & \\
 & 1 & e_2 & f_2 & & & & g_2 & & h_2 & \\
 & & \cdot & \cdot & \cdot & & & \cdot & & \cdot & \\
 & & & \cdot & \cdot & \cdot & & \cdot & & \cdot & \\
 & & & & \cdot & \cdot & \cdot & \cdot & & \cdot & \\
 & & & & & 1 & e_{N-4} & f_{N-4} & g_{N-4} & & h_{N-4} \\
 & & & & & & 1 & e_{N-3} & (f_{N-3} + g_{N-3}) & h_{N-3} & \\
 & & & & & & & 1 & (e_{N-2} + g_{N-2}) & (f_{N-2} + h_{N-2}) & \\
 & & & & & & & & 1 & (e_{N-1} + h_{N-1}) & \\
 & & & & & & & & & 1 &
\end{pmatrix}. \tag{5}$$

The elements of the matrices D and T are given by the following relations.

For the first $(N - 2)$ columns, the factorization is straightforward, as there is no interaction between the elements in T; so that

$$d_1 = (c_1)^{1/2}, \quad d_2 = (c_2 - b_1^2/d_1^2)^{1/2}, \quad e_1 = b_1/d_1\,d_2,$$

and for $i = 3, 4, \ldots, N - 2$

$$u = a_{i-2}/d_{i-2}; \quad v = b_{i-1}/d_{i-1} - e_{i-2}u$$

and

$$d_i = (c_i - u^2 - v^2)^{1/2}, \quad f_{i-2} = u/d_i, \quad e_{i-1} = v/d_i.$$

For the last two columns auxiliary vectors $\mathbf{x}$ and $\mathbf{y}$ are used; so that for column $(N - 1)$ if

$$u = a_{N-3}/d_{N-3}, \quad v = b_{N-2}/d_{N-2} - e_{N-3}u,$$

$$x_1 = a_{N-1}/d_1, \quad x_2 = -e_1x_1,$$

and for $i = 3, 4, \ldots, N - 2$

$$x_i = -e_{i-1}x_{i-1} - f_{i-2}x_{i-2},$$

then

$$d_{N-1} = \left\{c_{N-1} - \left[\sum_{i=1}^{N-4} x_i^2 + (x_{N-3} + u)^2 + (x_{N-2} + v)^2\right]\right\}^{1/2},$$

$$f_{N-3} = u/d_{N-1},$$

$$e_{N-2} = v/d_{N-1},$$

and for $i = 1, 2, \ldots, N-2$

$$g_i = x_i/d_{N-1}.$$

For the final column, we set

$$u = a_{N-2}/d_{N-2}, \quad v = b_{N-1}/d_{N-1} - e_{N-2}u,$$

$$y_1 = b_N/d_1, \quad y_2 = a_N/d_2 - e_1y_1,$$

and for $i = 3, 4, \ldots, N-1$

$$y_i = -e_{i-1}y_{i-1} - f_{i-2}y_{i-2},$$

and finally

$$y_{N-1} = y_{N-1} - \sum_{i=1}^{N-2} g_iy_i - g_{N-2}u.$$

Then

$$d_N = \{c_N - \sum_{i=1}^{N-3} y_i^2 - (y_{N-2} + u)^2 - (y_{N-1} + v)^2\}^{1/2},$$

$$f_{N-2} = u/d_N,$$

$$e_{N-1} = v/d_N,$$

and for $i = 1, 2, \ldots, N-1$

$$h_i = y_i/d_N.$$

The factorization stage is then complete. To solve the linear system (3), with coefficient matrix B as given by (1), we rewrite it in the equivalent form:

$$DT'TD\boldsymbol{\phi} = \mathbf{s} \quad \text{i.e.} \quad T'T(D\boldsymbol{\phi}) = D^{-1}\mathbf{s}\,; \tag{6}$$

so that putting

$$D\boldsymbol{\phi} = \mathbf{z}, \quad D^{-1}\mathbf{s} = \mathbf{q}\,, \tag{7}$$

the system becomes

$$T'T\mathbf{z} = \mathbf{q}\,,$$

which may be solved directly for $\mathbf{z}$ in terms of the auxiliary vector $\mathbf{p}$ whose elements are given by

$$p_1 = q_1, \quad p_2 = q_2 - e_1p_1\,,$$

and for $i = 3, 4, \ldots, N-2$

$$p_i = q_i - e_{i-1}\,p_{i-1} - f_{i-2}\,p_{i-2}\,.$$

Finally,

$$p_{N-1} = q_{N-1} - (e_{N-2} + g_{N-2})p_{N-2} - (f_{N-3} + g_{N-3})p_{N-3} - \sum_{i=1}^{N-4} g_ip_i\,,$$

$$p_N = q_N - (e_{N-1} + h_{N-1})p_{N-1} - (f_{N-2} + h_{N-2})p_{N-2} - \sum_{i=1}^{N-3} h_ip_i\,.$$

The elements of $\mathbf{z}$ are then obtained by a back substitution process so that

$$z_N = p_N\,,$$

$$z_{N-1} = p_{N-1} - (e_{N-1} + h_{N-1})z_N\,,$$

$$z_{N-2} = p_{N-2} - (e_{N-2} + g_{N-2})z_{N-1} - (f_{N-2} + h_{N-2})z_N\,,$$

and for $i = N - 3, \ldots, 3, 2, 1$

$$z_i = p_i - e_i z_{i+1} - f_i z_{i+2} - g_i z_{N-1} - h_i z_N .$$

The final solution $\boldsymbol{\phi}$ is then obtained from $\boldsymbol{\phi} = D^{-1}\mathbf{z}$ using only one division per component.

Storage Management Used in the Program

The algorithm has been written in the form of three subroutines, as follows.

1. FACTOR(A,B,C,G,H,N)

This subroutine performs the factorization stage. A matrix of the form (1) is input as three vectors **a**, **b**, **c**, which are overwritten during the course of the factorization. The diagonal elements of D, i.e. d_1 to d_n, replace the coefficients c_1 to c_n; in the vectors **b** and **a** are stored the elements e_1 to e_{n-1} and f_1 to f_{n-2}, respectively, which occur in the matrix T. In the factorization formulas two vectors **x** and **y** are used, but in the program these are replaced by **g** and **h** which are in turn overwritten by the g_1 to g_{n-2} and h_1 to h_{n-1} values which occur in the last two columns of T.

2. RHS(D,S,N)

This subroutine forms $D^{-1}\mathbf{s}$, where D is a diagonal matrix whose elements are stored in a vector **d**. The input vector **s** is overwritten by the result vector.

3. SOLVE(E,F,G,H,Q,N)

This subroutine solves the set of equations $T'T\mathbf{z} = \mathbf{q}$ for T as given by (5). The solution is effected in two stages, viz. a forward substitution for **p** where $T'\mathbf{p} = \mathbf{q}$ and a backward substitution for **z** where $T\mathbf{z} = \mathbf{p}$. The input vector **q** is overwritten first by the intermediate solution **p** and finally by the result **z**.

To solve (3) directly with B as given by (1) would then require the subroutine calls:

```
CALL FACTOR(A,B,C,G,H,N)
CALL RHS(C,S,N)
CALL SOLVE(B,A,G,H,S,N)
CALL RHS(C,S,N)
```

The reason for the three subroutines (instead of a single one), however, is that in a typical block successive overrelaxation process the three stages need to be performed at different points and possibly a different number of times. For instance, if the matrix B remains the same for each block, then the factorization need be performed only once. Moreover, the set of equations (3) is typically solved as part of an iterative procedure so that for block r, if k represents the iteration count then we have for successive block overrelaxation

$$B_r \boldsymbol{\phi}_r^{(k+1)} = \mathbf{s}(\boldsymbol{\phi}_{r-1}^{(k+1)}, \boldsymbol{\phi}_r^{(k)}, \boldsymbol{\phi}_{r+1}^{(k)}).$$

However, if $D_r \boldsymbol{\phi}_r^{(k)} = \mathbf{z}_r^{(k)}$, then the iteration is carried out in the transformed variable **z**, so that

$$T_r' T_r \mathbf{z}_r^{(k+1)} = \mathbf{q}(\mathbf{z}_{r-1}^{(k+1)}, \mathbf{z}_r^{(k)}, \mathbf{z}_{r+1}^{(k)})$$

until for some value K of k, $\mathbf{z}_r^{(K+1)}$ and $\mathbf{z}_r^{(K)}$ agree to some specified accuracy for all values of r. The final result is then obtained from $\boldsymbol{\phi} = D^{-1}\mathbf{z}$. The process is considered to be a normalized Choleski technique used in an inner loop where the need to include the division is eliminated until the final result is achieved.

Normalized Algorithm Versus Choleski Factorization

Let us briefly consider the differences between using the normalized factorization $B = DT'TD$ as opposed to a standard Choleski factorization $B = LL'$ which takes into account the structure of the nonzero elements of B.

The normalized form is clearly similar to a Choleski factorization since DT' is a lower triangular matrix, say $\tilde{L}$, so that $B = \tilde{L}\tilde{L}'$.

Moreover, if we consider $B = LL'$, then the formulas for determining the elements of L are very similar to those already given for the elements of D and T.

However, a work count of the two algorithms shows that the normalized version requires $2n$ more divisions than the Choleski form, this extra work occurring in the factorization stage. Having factorized B, the two solution processes for determining $\boldsymbol{\phi}$ from $B\boldsymbol{\phi} = \mathbf{s}$ require the same amount of work. Consequently, for the straightforward solution of a set of equations $B\boldsymbol{\phi} = \mathbf{s}$ with B as in (1), the Choleski factorization is preferable because less work is involved.

However, as we have seen, the B matrices are likely to occur in problems where the solution of $B\boldsymbol{\phi} = \mathbf{s}$ is incorporated into an iterative procedure. In this case for the normalized version, n divisions per block would be saved in each iteration in excess of one (since the iteration is effected in a transformed variable thus saving the final division). Let us consider all possible cases for the B matrices, i.e.

(i) $B = B_r^{(k)}$: the matrices change from block to block and with each iteration;
(ii) $B = B^{(k)}$: the matrices change with each iteration but not from block to block;
(iii) $B = B_r$: the matrices change from block to block but not with each iteration;
(iv) $B = B$: the matrices remain constant throughout the process.

Cases (iii) and (iv) are the most likely to occur in practice.

For (i), the Choleski factorization always involves less work.

For (ii), the Choleski form is more efficient if the number of blocks is less than three. For three or more blocks, the normalized form is more efficient if the number of iterations exceeds two.

For (iii) and (iv), i.e. the most likely cases, the normalized form is more efficient, from the point of view of number of operations involved, if the number of iterations is again more than two.

Consequently we conclude that for the majority of problems involving a block overrelaxation technique in which the blocks have a coefficient matrix of the form (1), the normalized algorithm as presented here would involve less work than a sparse Choleski form.

REFERENCES

1. Benson, A., and Evans, D.J. The successive peripheral block over-relaxation method. *J. Inst. Math. Appl. 9* (1972), 68–79.
2. Benson, A., and Evans, D.J. Algorithm 80: An algorithm for the solution of periodic quindiagonal systems of linear equations. *Computer J. 16*, 3 (1973), 278–279.
3. Cuthill, E.H., and Varga, R.S. A method of normalized block iteration. *J.ACM 6*, 2 (April 1959), 236–244.
4. Wood, W.L. Periodicity effects on the iterative solution of elliptic difference equations. *SIAM J. Numer. Anal. 8*, 2 (1971), 439–464.

ALGORITHM

```
      SUBROUTINE FACTOR(A, B, C, G, H, N)                                FAC   1Ø
C THE SUBROUTINE IS A NORMALISED FACTORISATION OF A SQUARE MATRIX OF      FAC   2Ø
C ORDER GREATER THAN 4.                                                  FAC   3Ø
C THE COEFFICIENT MATRIX P IS SYMMETRIC,POSITIVE DEFINITE AND            FAC   4Ø
C QUINDIAGONAL WITH NON-ZERO ELEMENTS ALSO IN THE LAST TWO COLUMNS OF    FAC   5Ø
C THE FIRST ROW, THE LAST COLUMN OF THE SECOND ROW, THE FIRST COLUMN      FAC   6Ø
C OF ROW (N-1) AND THE FIRST TWO COLUMNS OF THE LAST ROW. A,B,C ARE      FAC   7Ø
C VECTORS EACH OF N ELEMENTS CONTAINING RESPECTIVELY THE SUPER-SUPER-     FAC   8Ø
C DIAGONAL, SUPER-DIAGONAL AND DIAGONAL ELEMENTS OF P I.E. P(I,I+2),      FAC   9Ø
C P(I,I+1) AND P(I,I). THE MATRIX ELEMENTS P(1,N-1), P(1,N), P(2,N)      FAC  1ØØ
C ARE STORED IN A(N-1), B(N), A(N) RESPECTIVELY, AS ARE P(N-1,1),        FAC  11Ø
C P(N,1) AND P(N,2)                                                      FAC  12Ø
C THE MATRIX P IS FACTORISED INTO P=DRTD WHERE D IS A DIAGONAL MATRIX     FAC  13Ø
C AND T IS A REAL, UPPER TRIANGULAR MATRIX WITH UNIT DIAGONAL ELEMENTS   FAC  14Ø
C AND NON-ZERO ELEMENTS IN THE LAST TWO COLUMNS AND ON THE SUPER- AND    FAC  15Ø
C SUPER-SUPER-DIAGONALS. R DENOTES THE TRANSPOSE OF T.                   FAC  16Ø
C THE INPUT VECTORS ARE OVERWRITTEN DURING THE FACTORISATION STAGE SO     FAC  17Ø
C THAT VECTOR C CONTAINS THE DIAGONAL ELEMENTS OF D, VECTOR B CONTAINS   FAC  18Ø
```

```
C THE ELEMENTS  T(I,I+1) FOR I=1 TO N-3  AND VECTOR A CONTAINS THE      FAC  19Ø
C ELEMENTS  T(I,I+2)  FOR I=1 TO N-4.                                   FAC  2ØØ
C TWO RESULT VECTORS G,H ARE USED TO STORE THE ELEMENTS  T(I,N-1) FOR   FAC  21Ø
C I=1 TO N-4  AND  T(I,N) FOR I=1 TO N-3 RESPECTIVELY. THE ELEMENTS     FAC  22Ø
C T(N-3,N-1), T(N-2,N-1), T(N-2,N) AND T(N-1,N) ARE THEN GIVEN BY       FAC  23Ø
C A(N-3)+G(N-3), B(N-2)+G(N-2), A(N-2)+H(N-2), B(N-1)+H(N-1)            FAC  24Ø
C RESPECTIVELY.                                                         FAC  25Ø
      DIMENSION A(N), B(N), C(N), G(N), H(N)                            FAC  26Ø
      N1 = N - 1                                                        FAC  27Ø
      N2 = N - 2                                                        FAC  28Ø
      N3 = N - 3                                                        FAC  29Ø
      N4 = N - 4                                                        FAC  3ØØ
      C(1) = SQRT(C(1))                                                 FAC  31Ø
      V = B(1)/C(1)                                                     FAC  32Ø
      C(2) = SQRT(C(2)-V*V)                                             FAC  33Ø
      B(1) = V/C(2)                                                     FAC  34Ø
      DO 1Ø I=3,N2                                                      FAC  35Ø
        I1 = I - 1                                                      FAC  36Ø
        I2 = I - 2                                                      FAC  37Ø
        U = A(I2)/C(I2)                                                 FAC  38Ø
        V = B(I1)/C(I1) - B(I2)*U                                       FAC  39Ø
        C(I) = SQRT(C(I)-U*U-V*V)                                       FAC  4ØØ
        A(I2) = U/C(I)                                                  FAC  41Ø
        B(I1) = V/C(I)                                                  FAC  42Ø
   1Ø CONTINUE                                                          FAC  43Ø
      G(1) = A(N1)/C(1)                                                 FAC  44Ø
      G(2) = -B(1)*G(1)                                                 FAC  45Ø
      U = A(N3)/C(N3)                                                   FAC  46Ø
      V = B(N2)/C(N2) - B(N3)*U                                         FAC  47Ø
      DO 2Ø I=3,N2                                                      FAC  48Ø
        G(I) = -B(I-1)*G(I-1) - A(I-2)*G(I-2)                           FAC  49Ø
   2Ø CONTINUE                                                          FAC  5ØØ
      W = Ø.Ø                                                           FAC  51Ø
      DO 3Ø I=1,N4                                                      FAC  52Ø
        W = W + G(I)*G(I)                                               FAC  53Ø
   3Ø CONTINUE                                                          FAC  54Ø
      C(N1) = SQRT(C(N1)-W-(G(N3)+U)**2-(G(N2)+V)**2)                   FAC  55Ø
      W = 1.Ø/C(N1)                                                     FAC  56Ø
      A(N3) = U*W                                                       FAC  57Ø
      B(N2) = V*W                                                       FAC  58Ø
      DO 4Ø I=1,N2                                                      FAC  59Ø
        G(I) = G(I)*W                                                   FAC  6ØØ
   4Ø CONTINUE                                                          FAC  61Ø
      H(1) = B(N)/C(1)                                                  FAC  62Ø
      H(2) = A(N)/C(2) - B(1)*H(1)                                      FAC  63Ø
      U = A(N2)/C(N2)                                                   FAC  64Ø
      V = B(N1)/C(N1) - B(N2)*U                                         FAC  65Ø
      DO 5Ø I=3,N1                                                      FAC  66Ø
        H(I) = -B(I-1)*H(I-1) - A(I-2)*H(I-2)                           FAC  67Ø
   5Ø CONTINUE                                                          FAC  68Ø
      W = Ø.Ø                                                           FAC  69Ø
      DO 6Ø I=1,N2                                                      FAC  7ØØ
        W = W + G(I)*H(I)                                               FAC  71Ø
   6Ø CONTINUE                                                          FAC  72Ø
      H(N1) = H(N1) - W - G(N2)*U                                       FAC  73Ø
      W = Ø.Ø                                                           FAC  74Ø
      DO 7Ø I=1,N3                                                      FAC  75Ø
        W = W + H(I)*H(I)                                               FAC  76Ø
   7Ø CONTINUE                                                          FAC  77Ø
      C(N) = SQRT(C(N)-W-(H(N2)+U)**2-(H(N1)+V)**2)                     FAC  78Ø
      W = 1.Ø/C(N)                                                      FAC  79Ø
      A(N2) = U*W                                                       FAC  8ØØ
      B(N1) = V*W                                                       FAC  81Ø
      DO 8Ø I=1,N1                                                      FAC  82Ø
        H(I) = H(I)*W                                                   FAC  83Ø
   8Ø CONTINUE                                                          FAC  84Ø
      RETURN                                                            FAC  85Ø
      END                                                               FAC  86Ø


      SUBROUTINE SOLVE(E, F, G, H, Q, N)                                SOL   1Ø
C THE SUBROUTINE SOLVES THE SET OF N LINEAR EQUATIONS RTZ=Q WHERE T IS   SOL   2Ø
C AN UPPER TRIANGULAR MATRIX WITH UNIT ELEMENTS ON THE DIAGONAL AND R    SOL   3Ø
C DENOTES THE TRANSPOSE OF T. THE OTHER NON-ZERO ELEMENTS OF T OCCUR ON  SOL   4Ø
C THE SUPER-DIAGONAL I.E. T(I,I+1), THE SUPER-SUPER-DIAGONAL I.E.       SOL   5Ø
```

```
C T(I,I+2) AND IN THE LAST TWO COLUMNS I.E. T(I,N-1) AND T(I,N).          SOL   6Ø
C THE STORAGE IS SUCH THAT THE FOLLOWING ARE EQUIVALENT----               SOL   7Ø
C        T(I,I+1)   AND E(I)   FOR I=1 TO N-3                            SOL   8Ø
C        T(I,I+2)   AND F(I)   FOR I=1 TO N-4                            SOL   9Ø
C        T(I,N-1)   AND G(I)   FOR I=1 TO N-4                            SOL  1ØØ
C        T(I,N)     AND H(I)   FOR I=1 TO N-3                            SOL  11Ø
C        T(N-3,N-1) AND F(N-3)+G(N-3)                                    SOL  12Ø
C        T(N-2,N-1) AND E(N-2)+G(N-2)                                    SOL  13Ø
C        T(N-2,N)   AND F(N-2)+H(N-2)                                    SOL  14Ø
C        T(N-1,N)   AND E(N-1)+H(N-1)                                    SOL  15Ø
C THE SOLUTION IS EFFECTED BY A FORWARD SUBSTITUTION PROCESS FOLLOWED     SOL  16Ø
C BY A BACKWARD SUBSTITUTION AND THE INPUT VECTOR Q IS SUCCESSIVELY       SOL  17Ø
C OVERWRITTEN BY THE INTERMEDIATE SOLUTION (OF THE FORWARD ELIMINATION)   SOL  18Ø
C AND THEN THE FINAL SOLUTION.                                            SOL  19Ø
      DIMENSION E(N), F(N), G(N), H(N), Q(N)                              SOL  2ØØ
      N1 = N - 1                                                          SOL  21Ø
      N2 = N - 2                                                          SOL  22Ø
      N3 = N - 3                                                          SOL  23Ø
      Q(2) = Q(2) - E(1)*Q(1)                                             SOL  24Ø
      DO 1Ø I=3,N2                                                        SOL  25Ø
        Q(I) = Q(I) - E(I-1)*Q(I-1) - F(I-2)*Q(I-2)                       SOL  26Ø
   1Ø CONTINUE                                                            SOL  27Ø
      U = Ø.Ø                                                             SOL  28Ø
      V = Ø.Ø                                                             SOL  29Ø
      DO 2Ø I=1,N2                                                        SOL  3ØØ
        U = U + G(I)*Q(I)                                                 SOL  31Ø
        V = V + H(I)*Q(I)                                                 SOL  32Ø
   2Ø CONTINUE                                                            SOL  33Ø
      Q(N1) = Q(N1) - E(N2)*Q(N2) - F(N3)*Q(N3) - U                       SOL  34Ø
      Q(N) = Q(N) - (E(N1)+H(N1))*Q(N1) - F(N2)*Q(N2) - V                 SOL  35Ø
      Q(N1) = Q(N1) - (E(N1)+H(N1))*Q(N)                                  SOL  36Ø
      Q(N2) = Q(N2) - (E(N2)+G(N2))*Q(N1) - (F(N2)+H(N2))*Q(N)            SOL  37Ø
      DO 3Ø II=1,N3                                                       SOL  38Ø
        I = N3 - II + 1                                                   SOL  39Ø
        Q(I) = Q(I) - E(I)*Q(I+1) - F(I)*Q(I+2) - G(I)*Q(N1) -            SOL  4ØØ
     *   H(I)*Q(N)                                                       SOL  41Ø
   3Ø CONTINUE                                                            SOL  42Ø
      RETURN                                                              SOL  43Ø
      END                                                                 SOL  44Ø


      SUBROUTINE RHS(D, S, N)                                             RHS   1Ø
C THE SUBROUTINE FORMS Q WHERE BQ=S AND B IS A DIAGONAL MATRIX OF ORDER   RHS   2Ø
C N WHOSE NON-ZERO DIAGONAL ENTRIES ARE STORED IN A VECTOR D OF N         RHS   3Ø
C ELEMENTS. THE INPUT VECTOR S IS OVERWRITTEN BY THE RESULT Q             RHS   4Ø
      DIMENSION D(N), S(N)                                                RHS   5Ø
      DO 1Ø I=1,N                                                         RHS   6Ø
        S(I) = S(I)/D(I)                                                  RHS   7Ø
   1Ø CONTINUE                                                            RHS   8Ø
      RETURN                                                              RHS   9Ø
      END                                                                 RHS  1ØØ
```

ALGORITHM 513
Analysis of In-Situ Transposition [F1]

ESKO G. CATE and DAVID W. TWIGG
Boeing Computer Services, Inc.

Key Words and Phrases: transposition in place, matrix transposition, permutation
CR Categories: 4.49, 5.9, 5.30
Language: Fortran

DESCRIPTION

Algorithms for in-place matrix transposition make use of the cyclic structure of the transposition mapping. A clear introduction to the process was given in 1958 by Windley [6]. In 1968 Boothroyd presented ACM Algorithm 302 [1]. Then in 1970 Laflin and Brefner presented ACM Algorithm 380 which they showed to be significantly faster than Algorithm 302 [4].

In this paper a number of theoretical results describing the transposition process are derived. Algorithms 302 and 380 are described and applications of some of the theoretical results to accelerate both algorithms are described. Finally, experimental comparison of the performance of Algorithm 380 and the improved version is presented.

Theoretical Development

Fortran conventions are assumed. The $m \times n$ matrix A is stored by column in a vector $\bar{y}$ of adjacent storage locations. It is convenient to number components of $\bar{y}$ from 0 to $mn - 1$. Using this convention, matrix element $A(I, J)$ is stored in y_k, where $k = m(J - 1) + I - 1$. The transposition process moves the contents of y_k to y_p where $p = n(I - 1) + J - 1$. Thus the transposition process can be regarded as a permutation mapping acting on the indices of $\bar{y}$. The symbol π is used to denote this mapping, and the set of indices which π permutes is denoted by S. With this notation, $S = [0, 1, \ldots, mn - 1]$ and the formal equation $\pi(k) = p$ indicates that the contents of storage location y_k is moved to location y_p in the transposition process.

The theoretical material is organized in three sections: additional notation, development of general properties of π, and description of the fixed point properties of π.

Notation

Symbols used in this paper are as follows.

$C(x)$ Cycle of π containing element $x \in S$. $C(x)$ is generated by repeated application of π to x., i.e. $C(x) = [x, \pi(x), \pi^2(x), \ldots]$.

$L(x)$ Length of the cycle $C(x)$.

Received 11 September 1975 and 1 April 1976.

Author's address: Boeing Computer Services, Inc., P.O. Box 24346, Seattle, WA 98124.

$P(d)$ Number of cycles of length d.

$\mu(n)$ Möbius function; for n having prime factorization, $n = P_1^{e_1} \dots P_k^{e_k}$,

$$\mu(n) = \begin{cases} 1 & \text{if } n = 1, \\ (-1)^k & \text{if } e_1 = e_2 = \cdots = e_k = 1, \\ 0 & \text{otherwise.} \end{cases}$$

$[a, b]$ Greatest common divisor of a and b.

$d \mid u$ d is a divisor of u.

General Properties of π

An analytic expression for π is given in [1]. That is, for $x \in S$,

$$\pi(x) = \begin{cases} nx \bmod mn - 1 & \text{if } x \neq mn - 1, \\ mn - 1 & \text{if } x = mn - 1. \end{cases} \tag{1}$$

In this paper the symbols x and y represent any element of S except $mn - 1$, and all algebraic expressions involving these are assumed evaluated mod $mn - 1$.

That π is a linear mapping is obtained directly from (1). That is,

$$(ax + by) = a\pi(x) + b\pi(y). \tag{2}$$

This property is used to relate π to its fixed points.

The inverse mapping is used in both Algorithms 302 and 380 to traverse the cycles in the reverse direction.

THEOREM 1. $\pi^{-1}(y) = \begin{cases} my & \text{for } y \in s,\ y \neq mn - 1, \\ mn - 1 & \text{for } y = mn - 1. \end{cases}$

PROOF. $\pi^{-1}(\pi(y)) = \pi(\pi^{-1}(y)) = mny \bmod mn - 1 = ((mn - 1)y + y) \bmod mn - 1 = y$.

The transposition algorithms depend upon the cyclic structure of π. Two theorems are presented which describe the properties of these cycles. Theorem 2 shows how to find the length of the longest cycle (i.e. the cycles starting with $x = 1$, $A(2, 1)$). It also demonstrates that the length of any cycle must be a divisor of the length of the longest cycle. Theorem 3 provides a number theoretic formula which gives the number of cycles of any length. No application has been found for Theorem 3, but it may be of interest to other researchers. The interested reader is referred to [2] for the proof.

THEOREM 2. *The order of π is $L(1)$.*

PROOF. The order of a permutation mapping is the least common multiple of the lengths of its cycles [5]. Let $d = L(1)$. Then $n^d = 1 \bmod mn - 1$. It is sufficient to show, for any $y \in S$, that $\pi^d(y) = y$. If $y = mn - 1$, then the theorem is trivially true. For $y \neq mn - 1$, $\pi^d(y) = n^d y \bmod mn - 1 = y$.

THEOREM 3. *Assume that $k > 1$. The number of cycles of length k, $P(k)$, is given by $P(k) = k^{-1} \sum_{d \mid k} \mu(k/d)[n^d - 1, mn - 1]$.*

The number of fixed points ($k = 1$) is discussed in the next section.

Fixed Point Properties of π

It is mainly through the symmetry with respect to the fixed points that computational leverage is gained. The mapping always has at least two fixed points, 0 and $mn - 1$. If S has a midpoint, then the midpoint is also a fixed point. Formally,

THEOREM 4. *If m and n are odd, then $\frac{1}{2}(mn - 1)$ is a fixed point.*

PROOF. Let $n = 2k + 1$. By eq. (1), $\pi[\frac{1}{2}(mn - 1)] = \frac{1}{2}(2k + 1)(mn - 1) = \frac{1}{2}(mn - 1)$

Two formulas for calculation of the number of fixed points of π are now given. Let n_f represent the number of fixed points of π.

THEOREM 5. $n_f - 1 = [n - 1, mn - 1] = [m - 1, mn - 1]$, *and the fixed points are uniformly spaced at 0, d, $2d$, ..., where $d = (mn - 1)/(n_f - 1)$.*

PROOF. Because of eq. (1), all the fixed points of π except $mn - 1$ satisfy the congruence $x(n - 1) = 0 \bmod mn - 1$. The solutions of this congruence are 0, d, $2d, \dots$, where $d = (mn - 1)/[n - 1, mn - 1]$. (See [5, ch. 8].) Clearly π and π^{-1} have the same fixed points; so $n_f - 1 = [m - 1, mn - 1]$ by symmetry.

THEOREM 6. $n_f - 1 = [m - 1, n - 1]$.

PROOF. Let $d = [m - 1, n - 1]$. By Theorem 5, $(n_f - 1) \mid d$. Conversely, the identity $(n - 1)(m + 1) = mn - 1 + ((n - 1) - (m - 1))$ shows that $d \mid mn - 1$. This and Theorem 5 show that $d \mid (n_f - 1)$. Hence $d = n_f - 1$.

The mapping π is symmetric with respect to fixed points in the sense that for fixed point a, where $a \neq mn - 1$,

$$\pi(a \pm x) = a \pm \pi(x). \tag{3}$$

The fixed point 0 is usually the only fixed point satisfying eq. (3). That is, if $m \neq 1$ and $n \neq 1$, the probability that π has exactly two fixed points (0 and $mn - 1$) is the probability that $m - 1$ and $n - 1$ are relatively prime (Theorem 6). This probability is $6/\pi^2 = .608$ [3, ch. 18]. The symmetry with respect to 0 is applied in Algorithm 380 and extended in this paper.

When $a = 0$, the nontrivial part of eq. (3) is

$$\pi(-x) = -\pi(x). \tag{4}$$

It is evident from eq. (4) that $C(-x) = -C(x)$, and consequently that $L(x) = L(-x)$. Equation (4) was proved by Laflin and Brefner [4] by more complicated methods and was the basis for their improved algorithm. We call the symmetric cycle $C(-x)$ the companion cycle. The cycle and the companion cycle in some cases coincide. In this event the length of the cycle is an even integer, and x and $-x$ are separated by half the cycle. That is,

THEOREM 7. *$C(x) = C(-x)$ if and only if $-x = n^k x$, where $k = L(x)/2$.*

PROOF. Since $-x \in C(x)$ there is a smallest integer k such that $-x = n^k x$. This implies that $x = n^k(-x)$, which in turn implies that $x = n^k(n^k(x))$. Thus $L(x) \mid 2k$. Since $k < L(x)$, we must have $L(x) = 2k$. Conversely, if $n^k x = -x$, then $-x \in C(x)$ and $C(x) = C(-x)$.

Algorithm Improvement

Algorithms 302 and 380 are now described. For each algorithm, efficiency improvements based on the theoretical development in the previous section are discussed. Finally, timing comparisons are given.

Algorithm 302 [1]. For $i = 1, \ldots, mn - 1$, compute the index $j = \pi^{-1}(i)$ and exchange y_i and y_j, provided $j \geq i$. The case $j < i$ indicates that the element originally in y_j is now elsewhere following previous exchanges. Then search the cycle $C(j)$ for the first index $jr \geq i$ and exchange y_{jr} and y_i. It terminates when y_{mn-2} is filled.

Symmetry with respect to the fixed point 0 [eq. (4)] can be exploited to accelerate Algorithm 302. The idea is to process the array from both ends toward the middle simultaneously. That is, process element i and $-i$ mod $(mn - 1)$ concurrently. This reduces the time spent searching cycles. The revised algorithm is then:

1. Set $i = 0$.
2. Increment i. If $i > (mn/2 - 1)$, terminate. Otherwise, perform step 3 ($mn/2$ is truncated to integer value because of Theorem 4).
3. Compute j as follows: Let r be the smallest positive integer such that $i \leq \pi^{-r}(i) \leq mn - 1 - i$. Then $j = \pi^{-r}(i)$.
4. If $j = i$, return to step 2. Otherwise, exchange y_i and y_j and perform step 5.
5. If $j = mn - 1 - i$, return to step 2. Otherwise, perform the symmetric exchange of y_{mn-1-j} and y_{mn-1-i}. (If $j = mn - 1 - i$, this exchange was already performed in step 4.) Return to step 2.

Algorithm 380 [4]. First the number of fixed points is determined by testing each element in the array to see if $\pi(i) = i$. Then the array is scanned from lowest to nighest subscript. Each array subscript in turn is checked to see if that subscript is the smallest subscript in its cycle. Whenever such an element is found, the entire cycle is moved. If the companion cycle is distinct, then the companion cycle is moved after movement of the cycle is complete. The algorithm keeps track of the total number of elements moved and terminates when all (mn) elements have

been moved. It is interesting to note that Algorithm 380 works its way backwards around each cycle, using the inverse, saving itself a fetch and store for each element moved. Algorithm 380 can achieve greater efficiency if scratch storage is available. The scratch storage is utilized to keep track of the elements which have already been moved.

The improvements to Algorithm 380 are based again on the symmetry with respect to 0 and upon the calculations of the number of fixed points. Theorem 6 allows use of Euclid's algorithm to calculate the number of fixed points. Theorem 7 and eq. (4) allow the processing of a cycle and its companion concurrently. When the cycle and its companion coincide, the process is terminated halfway through the cycle (Theorem 7). We also used the linearity to calculate $\pi(x + 1) = \pi(x) + n$ to start the next cycle. It is interesting to note that on the computer used (CDC 6600 using the RUN compiler), the calculation of $mx \bmod mn - 1$ using the system modulo functions is slower than using the formulation used in Algorithm 380.

Timing Comparisons

The timing tests utilized a set of 21 test matrices which were presented in [4]. The per-element transposition time is defined to be the time required to transpose a matrix divided by the number of elements in the matrix. Each algorithm was tested on the complete set of test matrices and the average per-element transposi-

Table I

Algorithm	Average per-element transposition time (μs)
Algorithm 302	92.2
Algorithm 302 (revised)	67.1
Algorithm 380 (without scratch storage)	70.7
Algorithm 380 (with $(m+n)/2$ scratch storage)	46.9
Algorithm 380 (revised) (without scratch storage)	49.2
Algorithm 380 (revised) (with $(m+n)/2$ scratch storage)	28.5

Table II

	IWRK = (M+N)/2			IWRK = 1		
Size M×N	Original T1*	Revised T2*	2×(T1−T2)/(T1+T2)	Original T3*	Revised T4*	2×(T3−T4)/(T3+T4)
7 × 60	1.58	.84	.61	2.45	1.63	.40
7 × 70	1.71	.86	.66	2.52	1.62	.43
7 × 80	3.09	2.01	.42	4.15	2.96	.33
7 × 90	2.32	1.21	.63	5.01	3.52	.35
7 × 100	2.50	1.28	.65	3.88	2.46	.45
8 × 60	1.63	.84	.64	1.66	.87	.62
8 × 70	3.46	2.29	.41	4.94	3.6	.31
8 × 80	4.50	3.10	.37	6.69	5.02	.29
8 × 90	2.53	1.25	.68	2.55	1.22	.71
8 × 100	3.96	2.41	.49	7.28	5.13	.35
9 × 60	2.44	1.42	.53	4.12	2.85	.36
9 × 70	3.32	2.11	.45	5.65	4.02	.34
9 × 80	2.56	1.22	.71	2.49	1.24	.67
9 × 90	2.81	1.45	.64	4.13	2.66	.43
9 × 100	3.28	1.84	.56	7.44	5.43	.31
45 × 50	11.19	7.07	.45	19.39	14.20	.31
45 × 60	9.54	4.70	.68	9.22	4.66	.66
46 × 50	16.19	11.23	.36	25.49	19.57	.26
46 × 60	17.49	12.37	.34	27.96	21.57	.26
47 × 50	18.45	13.54	.31	25.90	20.37	.24
47 × 60	9.29	4.94	.61	9.65	4.89	.65

* Times are given in seconds/100.

tion time was calculated. These are summarized in Table I. A detailed comparison of the best algorithms (380 and the revised 380) is presented in Table II in the format utilized in [4]. All tests were performed using a CDC 6600, with compilation performed by the RUN compiler.

The simple modification of Algorithm 302 resulted in about a 25-percent efficiency improvement (Table I). The performance improvement of the revised Algorithm 380 varied from 20 percent to 70 percent on individual test matrices (Table II) with an average of about 35 percent.

REFERENCES

1. BOOTHROYD, J. Algorithm 302: Transpose vector stored array. *Comm. ACM 10*, 5 (1967), 292–293.
2. CATE, E.G., AND TWIGG, D.W. In place matrix transpose. Coord. Sheet NA-55, Boeing Computer Services, Seattle, Wash., May 19, 1975.
3. HARDY AND WRIGHT. *The Theory of Numbers*. Oxford U. Press, Oxford, fourth ed., 1962.
4. LAFLIN, S., AND BREFNER, M.A. Algorithm 380: In-situ transposition of a rectangular matrix, *Comm. ACM 13*, 5 (1970), 324–326.
5. MARSHALL, H., JR. *The Theory of Groups*. MacMillan, New York, 1959.
6. WINDLEY, P.F. Transposing matrices in a digital computer. *Comput. J. 2* (1959), 47–48.

ALGORITHM

```
      SUBROUTINE TRANS(A, M, N, MN, MOVE, IWRK, IOK)                    TRA   1Ø
C *****                                                                 TRA   2Ø
C  ALGORITHM 38Ø - REVISED                                              TRA   3Ø
C *****                                                                 TRA   4Ø
C  A IS A ONE-DIMENSIONAL ARRAY OF LENGTH MN=M*N, WHICH                 TRA   5Ø
C  CONTAINS THE MXN MATRIX TO BE TRANSPOSED (STORED                     TRA   6Ø
C  COLUMNWISE). MOVE IS A ONE-DIMENSIONAL ARRAY OF LENGTH IWRK          TRA   7Ø
C  USED TO STORE INFORMATION TO SPEED UP THE PROCESS.  THE              TRA   8Ø
C  VALUE IWRK=(M+N)/2 IS RECOMMENDED. IOK INDICATES THE                 TRA   9Ø
C  SUCCESS OR FAILURE OF THE ROUTINE.                                   TRA  1ØØ
C  NORMAL RETURN  IOK=Ø                                                 TRA  11Ø
C  ERRORS         IOK=-1 ,MN NOT EQUAL TO M*N                           TRA  12Ø
C                 IOK=-2 ,IWRK NEGATIVE OR ZERO                         TRA  13Ø
C                 IOK.GT.Ø, (SHOULD NEVER OCCUR),IN THIS CASE           TRA  14Ø
C  WE SET IOK EQUAL TO THE FINAL VALUE OF I WHEN THE SEARCH             TRA  15Ø
C  IS COMPLETED BUT SOME LOOPS HAVE NOT BEEN MOVED                      TRA  16Ø
C  NOTE * MOVE(I) WILL STAY ZERO FOR FIXED POINTS                       TRA  17Ø
      DIMENSION A(MN), MOVE(IWRK)                                       TRA  18Ø
C CHECK ARGUMENTS AND INITIALIZE.                                       TRA  19Ø
      IF (M.LT.2 .OR. N.LT.2) GO TO 12Ø                                 TRA  2ØØ
      IF (MN.NE.M*N) GO TO 18Ø                                          TRA  21Ø
      IF (IWRK.LT.1) GO TO 19Ø                                          TRA  22Ø
      IF (M.EQ.N) GO TO 13Ø                                             TRA  23Ø
      NCOUNT = 2                                                        TRA  24Ø
      K = MN - 1                                                        TRA  25Ø
      DO 1Ø I=1,IWRK                                                    TRA  26Ø
        MOVE(I) = Ø                                                     TRA  27Ø
   1Ø CONTINUE                                                          TRA  28Ø
      IF (M.LT.3 .OR. N.LT.3) GO TO 3Ø                                  TRA  29Ø
C CALCULATE THE NUMBER OF FIXED POINTS, EUCLIDS ALGORITHM               TRA  3ØØ
C FOR GCD(M-1,N-1).                                                     TRA  31Ø
      IR2 = M - 1                                                       TRA  32Ø
      IR1 = N - 1                                                       TRA  33Ø
   2Ø IRØ = MOD(IR2,IR1)                                                TRA  34Ø
      IR2 = IR1                                                         TRA  35Ø
      IR1 = IRØ                                                         TRA  36Ø
      IF (IRØ.NE.Ø) GO TO 2Ø                                            TRA  37Ø
      NCOUNT = NCOUNT + IR2 - 1                                         TRA  38Ø
C SET INITIAL VALUES FOR SEARCH                                         TRA  39Ø
   3Ø I = 1                                                             TRA  4ØØ
      IM = M                                                            TRA  41Ø
C AT LEAST ONE LOOP MUST BE RE-ARRANGED                                 TRA  42Ø
      GO TO 8Ø                                                          TRA  43Ø
C SEARCH FOR LOOPS TO REARRANGE                                         TRA  44Ø
   4Ø MAX = K - I                                                       TRA  45Ø
      I = I + 1                                                         TRA  46Ø
      IF (I.GT.MAX) GO TO 16Ø                                           TRA  47Ø
      IM = IM + M                                                       TRA  48Ø
      IF (IM.GT.K) IM = IM - K                                          TRA  49Ø
      I2 = IM                                                           TRA  5ØØ
```

```
      IF (I.EQ.I2) GO TO 4Ø                                    TRA  51Ø
      IF (I.GT.IWRK) GO TO 6Ø                                  TRA  52Ø
      IF (MOVE(I).EQ.Ø) GO TO 8Ø                               TRA  53Ø
      GO TO 4Ø                                                 TRA  54Ø
   5Ø I2 = M*I1 - K*(I1/N)                                     TRA  55Ø
   6Ø IF (I2.LE.I .OR. I2.GE.MAX) GO TO 7Ø                     TRA  56Ø
      I1 = I2                                                  TRA  57Ø
      GO TO 5Ø                                                 TRA  58Ø
   7Ø IF (I2.NE.I) GO TO 4Ø                                    TRA  59Ø
C REARRANGE THE ELEMENTS OF A LOOP AND ITS COMPANION LOOP      TRA  6ØØ
   8Ø I1 = I                                                   TRA  61Ø
      KMI = K - I                                              TRA  62Ø
      B = A(I1+1)                                              TRA  63Ø
      I1C = KMI                                                TRA  64Ø
      C = A(I1C+1)                                             TRA  65Ø
   9Ø I2 = M*I1 - K*(I1/N)                                     TRA  66Ø
      I2C = K - I2                                             TRA  67Ø
      IF (I1.LE.IWRK) MOVE(I1) = 2                             TRA  68Ø
      IF (I1C.LE.IWRK) MOVE(I1C) = 2                           TRA  69Ø
      NCOUNT = NCOUNT + 2                                      TRA  7ØØ
      IF (I2.EQ.I) GO TO 11Ø                                   TRA  71Ø
      IF (I2.EQ.KMI) GO TO 1ØØ                                 TRA  72Ø
      A(I1+1) = A(I2+1)                                        TRA  73Ø
      A(I1C+1) = A(I2C+1)                                      TRA  74Ø
      I1 = I2                                                  TRA  75Ø
      I1C = I2C                                                TRA  76Ø
      GO TO 9Ø                                                 TRA  77Ø
C FINAL STORE AND TEST FOR FINISHED                            TRA  78Ø
  1ØØ D = B                                                    TRA  79Ø
      B = C                                                    TRA  8ØØ
      C = D                                                    TRA  81Ø
  11Ø A(I1+1) = B                                              TRA  82Ø
      A(I1C+1) = C                                             TRA  83Ø
      IF (NCOUNT.LT.MN) GO TO 4Ø                               TRA  34Ø
C NORMAL RETURN                                                TRA  85Ø
  12Ø IOK = Ø                                                  TRA  86Ø
      RETURN                                                   TRA  87Ø
C IF MATRIX IS SQUARE,EXCHANGE ELEMENTS A(I,J) AND A(J,I).     TRA  88Ø
  13Ø N1 = N - 1                                               TRA  89Ø
      DO 15Ø I=1,N1                                            TRA  9ØØ
        J1 = I + 1                                             TRA  91Ø
        DO 14Ø J=J1,N                                          TRA  92Ø
          I1 = I + (J-1)*N                                     TRA  93Ø
          I2 = J + (I-1)*M                                     TRA  94Ø
          B = A(I1)                                            TRA  95Ø
          A(I1) = A(I2)                                        TRA  96Ø
          A(I2) = B                                            TRA  97Ø
  14Ø   CONTINUE                                               TRA  98Ø
  15Ø CONTINUE                                                 TRA  99Ø
      GO TO 12Ø                                                TRA 1ØØØ
C ERROR RETURNS.                                               TRA 1Ø1Ø
  16Ø IOK = I                                                  TRA 1Ø2Ø
  17Ø RETURN                                                   TRA 1Ø3Ø
  18Ø IOK = -1                                                 TRA 1Ø4Ø
      GO TO 17Ø                                                TRA 1Ø5Ø
  19Ø IOK = -2                                                 TRA 1Ø6Ø
      GO TO 17Ø                                                TRA 1Ø7Ø
      END                                                      TRA 1Ø8Ø
```

ACM Transactions on Mathematical Software, Vol. 5, No. 4, December 1979, Page 520.

REMARK ON ALGORITHM 513

Analysis of In-Situ Transposition [F1]
[E.G. Cate and D.W. Twigg, *ACM Trans. Math. Software 3,* 1 (March 1977), 104-110]

and

REMARK ON ALGORITHM 467

Matrix Transposition in Place [F1]
[N. Brenner, *Comm. ACM 16,* 11 (Nov. 1973), 692-694]

Burton L. Leathers [Recd 8 March 1978 and 19 May 1978]
Faculty of Human Kinetics and Leisure Studies, University of Waterloo, Waterloo, Ont., Canada

It is somewhat distressing to note the publication of Algorithm 513. While the algorithm is, as claimed, an improvement over Algorithm 380 [1], it is by no means as good as Algorithm 467, which was published long before Algorithm 513 was even submitted for publication. A reanalysis of the timing figures in Algorithms 467 and 513 suggests very strongly that Algorithm 467 is superior to Algorithm 513 except, possibly, when the modulus of the generating function for the transposition permutation is prime.

A direct comparison of the algorithms was performed using the Fortran G (nonoptimized) compiler on an IBM 370/158 running under VM/CMS. The results, reported in Table I, indicate quite clearly that Algorithm 467 enjoys a nearly 2:1 advantage except when the modulus is prime. It should be noted that it is not so much that Algorithm 467 is degraded in this case as that Algorithm 513 performs exceptionally well.

Table I. Timing Results for Algorithms 467 and 513
(All times are in milliseconds. Work area sizes are $(R + C)/2$ in all cases.)

Rows	Columns	Modulus	467	513	513/467
45	50	13.173	58	88	1.52
45	60	2699	72	68	.94
45	180	7.13.89	212	444	2.08
45	200	8999	214	199	.93
46	50	$11^2 19$	58	128	2.21
46	60	31.89	72	147	2.04
46	180	17.487	213	359	1.69
46	200	9199	212	181	.85
47	50	34.29	54	134	2.46
47	60	2819	66	62	.94
47	180	11.769	207	411	1.99
47	200	2.13.241	430	628	1.46

REFERENCES

1. LAFLIN, S., AND BREBNER, M.A. Algorithm 380. In-situ transposition of a rectangular matrix. *Comm. ACM 13,* 5 (May 1970), 324-326.

ALGORITHM 514
A New Method of Cubic Curve Fitting Using Local Data [E2]

T. M. R. ELLIS and D. H. McLAIN
University of Sheffield, England

Key Words and Phrases: interpolation, cubic splines, local data
CR Categories: 5.13
Language: Algol

DESCRIPTION

This algorithm is a local interpolation method to find an approximating function $y = f(x)$ through a set of given data points (x_i, y_i) with $x_1 < \cdots < x_i < x_{i+1} < \cdots$. Probably the best known methods for this are cubic splines [3, 6, 7] and the "classical" Lagrangian polynomial interpolation, of which Aitken's method is perhaps the most stable implementation [2]. Both of these methods are nonlocal, i.e. the interpolated values depend on all the data points. Under many circumstances, however, it is desirable to use a local interpolation method for which the values of the interpolating function depend only on a few of the nearest data points. For example, a local method may be preferred in an interactive system where one does not want to recalculate a complete picture when one point is changed, or where one wants to save storage in multidimensional interpolation.

The method used in our algorithm is similar to cubic spline interpolation in that different cubic, or third degree, polynomials $y = f_i(x)$ represent the function between different pairs of data points $(x_i, y_i) - (x_{i+1}, y_{i+1})$ and in that the coefficients are chosen to ensure continuity of the function and its first derivative at each data point. We have, however, sacrificed the spline's continuity of the second derivative in order to ensure that the coefficients depend only on the local data points.

A similar approach is described by Maude [4] which also ensures the continuity of the function and its first derivative at each data point. However, Maude's functions, at their simplest, reduce to quintic (fifth degree) polynomials, whereas ours are cubic polynomials with resulting computational advantages. Another method with a similar goal is due to Akima [1]. But whereas Akima's method is designed from geometrical considerations, ours is algebraic and appears to have some practical advantages. Thus our method gives an exact fit to the original polynomial $f(x)$ if the data points satisfy a cubic equation $y_i = f(x_i)$. As a result, in general it is more accurate for data of mathematical origin, or where smooth curves are expected. Our algorithm also appears to give more spline-like functions than Akima's, and in particular tends to produce much smaller discontinuities in

Received 20 November 1974 and 30 April 1976.

Authors' address: Computing Services, University of Sheffield, Sheffield S10 2TN, England.

the second derivatives at the data points. This is particularly important when interpolating surfaces, $z = f(x, y)$, by interpolating successively in two independent directions, x and y; for when depicting a perspective view of the surface, for example, a discontinuity in the second derivative of the function often results in a noticeable irregularity (a kink) in a visible edge.

Tables I and II compare the performance of our algorithm with other interpolation methods, and give the results obtained from a program run on a Control Data 7600 which applies six methods to data generated from five mathematical functions. Table I shows the average deviation found between the interpolated result and the true value from the original function, while Table II gives the average discontinuity of the second derivative at the nodes. The data used for the interpolations were generated for the following 18 unequally spaced values of x: -2.95, -2.6, -2.1, -1.8, -1.4, -1.0, -0.75, -0.3, -0.05, 0.2, 0.55, 0.9, 1.25, 1.6, 1.7, 2.1, 2.4, 3.0.

The first three methods are nonlocal ones: Aitken's method and two cubic spline methods. The first of the two cubic splines (a) is as implemented in Späth [7] and is probably the most used of the spline routines in the NAG Library [5], while the second cubic spline (b) was implemented by Reid [6]. They differ mainly in their treatment of the end intervals: Späth sets the second derivative at the endpoints to zero; Reid forces the third derivative at the points adjacent to the ends to be continuous. The second three methods given in Table I are local ones: Akima's, Maude's, and the method described below.

Our method is implemented in the two procedures described here. For each point (x_i, y_i) the first procedure, *find gradients at nodes*, initially finds that cubic poly-

Table I. Average Deviation from Original Function for Various Interpolation Methods ($\times 10^6$)

	Method					
	Nonlocal			Local		
Function	Aitken's	Cubic splines (a)	Cubic splines (b)	Akima's	Maude's	Present
x^3	0	28051	0	25709	8172	0
x^4	0	165921	4561	112169	68566	6434
$e^{-x^2/2}$	36	164	76	1260	640	78
tanh x	10409	140	147	833	619	214
sin x	0*	297	116	1531	1055	176

* The actual value was 0.011 ($\times 10^{-6}$). Aitken's method gives a remarkably accurate fit for sin x and a remarkably poor one for tanh x.

Table II. Average Discontinuity of Second Derivative at Data Points ($\times 10^4$)

	Method			
	All nonlocal methods as in	Local		
Function	Table I	Akima's	Maude's	Present
x^3	0	79953	0	0
x^4	0	391953	0	5766
$e^{-x^2/2}$	0	5027	0	268
tanh x	0	4844	0	469
sin x	0	6293	0	198

nomial which passes through the three points (x_{i-1}, y_{i-1}), (x_i, y_i), (x_{i+1}, y_{i+1}) and which gives a least squares fit to the two neighboring data points (x_{i-2}, y_{i-2}), (x_{i+2}, y_{i+2}). As written, in the least squares fitting the weights given to the last two points are not equal, but are inversely proportional to the squares of their distances from x_{i-1} and x_{i+1}, respectively; we have found experimentally that such weighting leads to smaller discontinuities in the second derivatives at the nodes. The gradient of this cubic at the central point (x_i, y_i) is then calculated. This process is repeated for all the points (x_i, y_i), except for the first two and the last two data points for which the subscripts x_{i-2} etc. or x_{i+2} etc. would be out of range; for these extreme points a cubic is fitted exactly through the first four or the last four points as appropriate.

The second procedure, *coeffs*, then constructs as the approximating function for the interval $x_i \leq x \leq x_{i+1}$ that cubic polynomial which has the correct ordinates y_i and y_{i+1} and the calculated gradients at the two endpoints. The procedure calculates the four coefficients c_0, c_1, c_2, c_3 for which the approximating polynomial may be expressed in the form

$$f(x) = c_0 + c_1(x - x_i) + c_2(x - x_i)^2 + c_3(x - x_i)^3.$$

This is preferred to the form $a_0 + a_1x + a_2x^2 + a_3x^3$ which is more unstable against roundoff errors.

Normally the procedure *find gradients at nodes* need only be called once for any given set of x_i, y_i, thus avoiding unnecessary repetition of calculation; the gradients it calculates are then used by the procedure *coeffs* for any required interval.

We have also found the method useful in two-dimensional interpolation to fit a surface $z = f(x, y)$ through data on a rectangular grid (x_i, y_j, z_{ij}). This may be achieved by first applying the method to all the rows of data, keeping y_j fixed and varying i, and then by applying it again to the resulting four sets of coefficients, but this time keeping x_i fixed and varying j.

ACKNOWLEDGMENTS

We would like to thank M.J.D. Powell and the referees for a number of valuable suggestions. We also would like to acknowledge the assistance provided by Mrs. S. Costello in the computation of the various figures shown in the tables.

REFERENCES

1. AKIMA, H. Interpolation and smooth curve fitting based on local procedures. *Comm. ACM 15*, 10 (Oct. 1972), 914–918.
2. FRÖBERG, C-E. *Introduction to Numerical Analysis*. Addison-Wesley, Reading, Mass., 1966, pp. 150–157.
3. GREVILLE, T.N.E. Spline Functions, interpolation, and numerical quadrature. In *Mathematical Methods for Digital Computers, Vol. 2*, Ralston A. and Wilf H. S. (Eds.) John Wiley, New York, 1967.
4. MAUDE, A.D. Interpolation—mainly for graph plotters. *Computer J. 16*, 1 (Feb. 1973), 64–65.
5. Numerical Algorithms Group. Subroutine E01ADF in *NAG Library Manual*. Numerical Algorithms Group, Oxford, England, 1973.
6. REID, J.K. Subroutine TB04A in *Harwell Subroutine Library*, M. J. Hopper (compiler), Rep. AERE-R6912, Atomic Energy Research Establishment, Harwell, England, 1971.
7. SPÄTH, H. Spline interpolation of degree three (Algorithm 40). *Computer J. 12*, 2 (May 1969), 198–199.

ALGORITHM

```
procedure find gradients at nodes (x,y,grad,lbx,ubx);
  value lbx,ubx; integer lbx,ubx;
  array x,y,grad;
comment x and y are arrays with subscripts from lbx
  to ubx giving the abscissae and ordinates respec-
  tively of the data points. These should be in
  ascending order of abscissae, with no two abscissae
```

```
    equal.  The arrays must contain at least four
    points.
    grad is an array having the same subscripts as x
    and y into which will be placed the calculated
    gradients of the required cubics at each of the
    data points.
    lbx and ubx are integers giving the minimum and
    maximum values respectively of the subscripts of
    the arrays x, y, and grad;
begin
  integer i,iless2,iless1,iplus1,iplus2;
  real x0,x1,x2,x3,x4,y2,prod1,prod2,num,denom,g,
    coeff2,xdiff,xprod,weight;
  for i:= lbx step 1 until ubx do
  begin
comment special treatment is needed at end points;
    iless1:= if i > lbx then i-1 else i+3;
    iplus1:= if i < ubx then i+1 else i-3;
    x2:= x[i];    y2:= y[i];
    x1:= x[iless1]-x2;    x3:= x[iplus1]-x2;
comment first fit a quadratic through x1,x2,x3;
    prod1:= (y[iless1]-y2)*x3;
    prod2:= (y[iplus1]-y2)*x1;
    denom:= x1*x3*(x[iless1]-x[iplus1]);
    g:= (x1*prod2-x3*prod1)/denom;
    coeff2:= (prod1-prod2)/denom;
comment if x0 exists, find its contribution to the
  cubic adjustment;
    if i ≤ lbx+1 then num:= denom:= 0.0 else
    begin
      iless2:= i-2;    x0:= x[iless2]-x2;
      xdiff:= x[iless2]-x[iless1];
      xprod:= x0*xdiff*(x[iless2]-x[iplus1]);
      weight:= xprod/(xdiff*xdiff);
      num:= weight*(y[iless2]-y2-x0*(g+x0*coeff2));
      denom:= weight*xprod
    end;
comment if x4 exists, find its contribution to the
  cubic adjustment;
    if i < ubx-1 then
    begin
      iplus2:= i+2;    x4:= x[iplus2]-x2;
      xdiff:= x[iplus2]-x[iplus1];
      xprod:= x4*xdiff*(x[iplus2]-x[iless1]);
      weight:= xprod/(xdiff*xdiff);
      num:= num+weight*(y[iplus2]-y2-x4*(g+x4*coeff2));
      denom:= denom+weight*xprod
    end;
    grad[i]:= g+num*x1*x3/denom
  end
end of procedure find gradients at nodes;

procedure coeffs(x,y,grad,i,c0,c1,c2,c3);
  value i;  integer i;
  real c0,c1,c2,c3;
  array x,y,grad;
```

```
comment This procedure calculates the coefficients
  of the cubic which has values y[i], y[i+1] and
  gradients grad [i], grad[i+1] at x[i], x[i+1]
  respectively.
  The cubic takes the form
    c0 + c1×(x-x[i]) + c2×(x-x[i])↑2 + c3×(x-x[i])↑3.
  To interpolate as suggested in the Description,
  above, these coefficients would be used for
  values of x in the range x[i]≤x≤x[i+1];
begin
  real h,dy;
  c0:= y[i];
  h:= x[i+1]-x[i];     dy:= y[i+1]-c0;
  c1:= grad[i];
  c2:= (3.0×dy-h×(2.0×c1+grad[i+1]))/(h×h);
  c3:= (h×(c1+grad[i+1])-2.0×dy)/h↑3
end of procedure coeffs;
```

ALGORITHM 515
Generation of a Vector from the Lexicographical Index [G6]

B. P. BUCKLES AND M. LYBANON
Computer Sciences Corporation

Key Words and Phrases: combinations
CR Categories: 5.39
Language: Fortran

DESCRIPTION

Let $C = \{C_1, C_2, \ldots, C_m\}$ be the set of combinations of n items taken p at a time and arranged in lexicographical order. Given an integer, i ($1 \leq i \leq m$, $n \geq p > 0$), the algorithm derives C_i. Previous algorithms [1, 3, 6] have accomplished this by the generation of all vectors between an initial point in the combination space and the desired vector. The cited algorithms are computationally advantageous if all combinations or a sequential subset are required. However, the algorithm given here is advantageous if a few randomly selected combinations are needed and each selection is not based on the previous selection history.

A one-to-one correspondence between the universe of n items and the first n natural numbers is established. The combinations produced by the algorithm are selections of p of the first n integers in lexicographical order [2, 5]. Thus the combinations produced by the algorithm may be regarded equivalently as pointers to combinations of any objects. If the combinations are generated sequentially (according to the index), exactly the same order is achieved as that of Mifsud's algorithm [4].

Vector generation is performed by probing the combination space and sequentially reducing the interval of uncertainty within which the indexed vector resides. Beginning with the leftmost vector position, trial values in ascending order for the digits are tested by computing the index of the (partial) trial vector, using essentially the method of Walter [7]. The lexicographical index, i, for a combination C_i is computed as follows: Let the elements of the combination vector be denoted $X_1, X_2, \ldots, X_p$. Using $X_0 = 0$, the index i is

$$i = 1 + \sum_{j=1}^{p} Q_j; \quad Q_j = \begin{cases} \sum_{k=X_{j-1}+1}^{X_j-1} \binom{n-k}{p-j} & \text{if } X_{j-1} + 1 \leq X_j - 1 \\ 0 & \text{otherwise.} \end{cases}$$

Received 24 March 1975, 14 January 1976, and 10 May 1976.

This work was supported by NASA under Contract NAS8-21805 for the Data Systems Laboratory, NASA—George C. Marshall Space Flight Center.
Authors' present addresses: B.P. Buckles, Science Applications, Inc., 2109 W. Clinton St., Huntsville, AL 35805; M. Lybanon, Computer Sciences Corporation, NSTL Support Operations, NSTL Station, MS 39529.

This formula is the inverse of the present algorithm. The technique consists of accumulating terms of the form of those in the expression for Q_j until a total greater than or equal to the desired vector index is reached. The value causing this condition is X_j. The sum is reduced by the amount of the last term (so that Q_j only contains terms through $X_j - 1$) and the search resumes at the next position to the right. The probing halts when there are no positions remaining. A previous ACM algorithm [8] is used to compute the binomial coefficients.

Once a value has been eliminated from a position, it is eliminated from all following positions since the values are produced in ascending order. Thus the number of probes required to find a vector is exactly the numeric value of the last element. That is, the vector {1, 3, 7} will be found in seven probes given any combination space containing seven or more things taken three at a time. This leads to an expression for the average number of probes required to find a vector based entirely on the frequency of values occurring as the last element:

$$\binom{n}{p}^{-1} \sum_{m=0}^{n-p} \binom{n-1-m}{p-1} (n - m).$$

It can be seen that each nonzero term in Q_p (i.e. Q_j for $j = p$) equals unity. If the first term (1) in the expression for i is combined with Q_p, it can be written in the form

$$Q_p = X_j - X_{j-1}.$$

Taking advantage of this identity, an improvement in performance may be obtained by computing

$$C_p = C_{p-1} + L - K,$$

where L and K are variables given within the algorithm.

ACKNOWLEDGMENT

The authors wish to thank the referees for their thought-provoking and helpful comments. In particular, a remark of one of the referees prompted an observation leading to the improvement expressed in the last equation.

REFERENCES

1. Chase, P.J. Algorithm 382: Combinations of M out of N objects. *Comm. ACM 13*, 6 (June 1970), 368.
2. Lehmer, D.H. The machine tools of combinatorics. In *Applied Combinatorial Mathematics*, E.F. Beckenbach, Ed., Wiley, New York, 1964, pp. 5-30.
3. Liu, C.M., and Tang, D.T. Algorithm 452: Enumerating combinations of M out of N objects. *Comm. ACM 16*, 8 (Aug. 1973), 485.
4. Mifsud, C.J. Algorithm 154: Combination in lexicographical order. *Comm. ACM 6*, 3 (March 1963), 103.
5. Nijenhuis, A., and Wilf, H.S. *Combinatorial Algorithms*. Academic Press, New York, 1975.
6. Phillips, J.P.N. Permutations of the elements of a vector in lexicographic order. *Comput. J. 10*, 4 (Oct. 1967), 311.
7. Walter, H.F. Algorithm 151: Location of a vector in a lexicographically ordered list. *Comm. ACM 6*, 2 (Feb. 1963), 68.
8. Wolfson, M.L., and Wright, H.V. Algorithm 160: Combinatorial of M things taken N at a time. *Comm. ACM 6*, 4 (April 1963), 161.

ALGORITHM

```
      SUBROUTINE COMB(N, P, L, C)                              COM  1Ø
C THIS SUBROUTINE FINDS THE COMBINATION SET OF N THINGS        COM  2Ø
C TAKEN P AT A TIME FOR A GIVEN LEXICOGRAPHICAL INDEX.         COM  3Ø
C N - NUMBER OF THINGS IN THE SET                              COM  4Ø
C P - NUMBER OF THINGS IN EACH COMBINATION                     COM  5Ø
C L - LEXICOGRAPHICAL INDEX OF COMBINATION SOUGHT              COM  6Ø
C C - OUTPUT ARRAY CONTAINING THE COMBINATION SET              COM  7Ø
```

```
C THE FOLLOWING RELATIONSHIPS MUST EXIST AMONG THE INPUT              COM   8Ø
C VARIABLES.  L MUST BE GREATER THAN OR EQUAL TO 1 AND LESS           COM   9Ø
C THAN OR EQUAL TO THE MAXIMUM LEXICOGRAPHICAL INDEX.                 COM  1ØØ
C P MUST BE LESS THAN OR EQUAL TO N AND GREATER THAN ZERO.            COM  11Ø
      INTEGER N, P, L, C(P), K, R, P1, BINOM                          COM  12Ø
C INITIALIZE LOWER BOUND INDEX AT ZERO                                COM  13Ø
      K = Ø                                                           COM  14Ø
C LOOP TO SELECT ELEMENTS IN ASCENDING ORDER                          COM  15Ø
      P1 = P - 1                                                      COM  16Ø
      DO 2Ø I=1,P1                                                    COM  17Ø
C SET LOWER BOUND ELEMENT NUMBER FOR NEXT ELEMENT VALUE               COM  18Ø
        C(I) = Ø                                                      COM  19Ø
        IF (I.NE.1) C(I) = C(I-1)                                     COM  2ØØ
C LOOP TO CHECK VALIDITY OF EACH ELEMENT VALUE                        COM  21Ø
   1Ø   C(I) = C(I) + 1                                               COM  22Ø
        R = BINOM(N-C(I),P-I)                                         COM  23Ø
        K = K + R                                                     COM  24Ø
        IF (K.LT.L) GO TO 1Ø                                          COM  25Ø
        K = K - R                                                     COM  26Ø
   2Ø CONTINUE                                                        COM  27Ø
      C(P) = C(P1) + L - K                                            COM  28Ø
      RETURN                                                          COM  29Ø
      END                                                             COM  3ØØ

      INTEGER FUNCTION BINOM(M, N)                                    BIN   1Ø
C ACM ALGORITHM 16Ø TRANSLATED TO FORTRAN.  CALCULATES THE            BIN   2Ø
C NUMBER OF COMBINATIONS OF M THINGS TAKEN N AT A TIME.               BIN   3Ø
      INTEGER M, N, P, I, N1, R                                       BIN   4Ø
      N1 = N                                                          BIN   5Ø
      P = M - N1                                                      BIN   6Ø
      IF (N1.GE.P) GO TO 1Ø                                           BIN   7Ø
      P = N1                                                          BIN   8Ø
      N1 = M - P                                                      BIN   9Ø
   1Ø R = N1 + 1                                                      BIN  1ØØ
      IF (P.EQ.Ø) R = 1                                               BIN  11Ø
      IF (P.LT.2) GO TO 3Ø                                            BIN  12Ø
      DO 2Ø I=2,P                                                     BIN  13Ø
        R = (R*(N1+I))/I                                              BIN  14Ø
   2Ø CONTINUE                                                        BIN  15Ø
   3Ø BINOM = R                                                       BIN  16Ø
      RETURN                                                          BIN  17Ø
      END                                                             BIN  18Ø
```

ALGORITHM 516
An Algorithm for Obtaining Confidence Intervals and Point Estimates Based on Ranks in the Two-Sample Location Problem [G1]

J. W. McKEAN and T. A. RYAN, JR.
Pennsylvania State University

Key Words and Phrases: asymptotic linearity, confidence interval, Illinois method, point estimate, ranks, regula falsi, robust, Wilcoxon-Mann-Whitney rank test
CR Categories: 5.5
Language: Fortran

DESCRIPTION

Suppose we are sampling from two populations which have the same continuous distribution except for a possible shift in location. For notational convenience let $X_1, \ldots, X_m$ be a random sample from $F(x)$ and let $Y_1, \ldots, Y_n$ be a random sample from $F(x - \theta)$. In testing hypotheses concerning θ the robust two-sample rank test (Wilcoxon-Mann-Whitney) is often used, but the related point estimate and confidence intervals are used relatively rarely. The reasons for this are primarily computational. While it has been practical to carry out the test for large sample sizes, it has not been practical for the estimates.

The most common method for computing the estimates is based on the $m \cdot n$ ordered differences $D_{ij} = Y_j - X_i$ (for instance, see Noether [5, p. 138]). These differences can be quickly obtained and ordered for small m and n; however, for large m and n they may be costly to obtain, even with the use of a computer. The storage is high (for example, if m and n are 1000, then 1,000,000 words of storage are needed); furthermore, computation time is high (of order $m \cdot n$, even if the samples are ordered in a preliminary step).

One way of avoiding the computation difficulty is to modify the estimate, for instance, as Antille [1] did in one sample problem. We propose instead an iteration procedure based on the function $U(\theta) = \#(Y_j - X_i \leq \theta)$. This is the Mann-Whitney test statistic for testing that the shift is θ; it differs only by a constant from the Wilcoxon rank sum statistic. It is an increasing function of θ which steps up one unit at each difference $Y_j - X_i$. The point estimate and confidence interval can be found by essentially inverting this function; for example, the point estimate of θ is $\hat{\theta}$ = median (D_{ij}), which can be found by solving the equation $U(\theta) = mn/2$ (the middle such value if it is not unique). The endpoints of the

Received 22 August 1975 and 10 May 1976.

Authors' present addresses: J.W. McKean, Department of Mathematical Sciences, University of Texas at Dallas, 2601 N. Floyd Rd., Richardson, TX 75080; T.A. Ryan, Jr., Department of Statistics, College of Science, Pennsylvania State University, 219 Pond Laboratory; University Park, PA 16802.

confidence interval can be obtained similarly. See Hettmansperger and McKean [3] for a graphical view of $U(\theta)$.

The algorithm given here finds the point estimate and confidence interval by solving such equations. Since $U(\theta)$ is monotone, methods such as regula falsi (linear interpolation) may be used. Of the methods tried, we have found the Illinois method, a modification of the regula falsi method outlined by Dowell and Jarratt [2], to be the best. The number of iterations is reduced by a factor of 2 or 3 when it is compared to regula falsi; hence we have incorporated it into the algorithm. The algorithm requires a preliminary and separate ordering of the X and Y. We have found that this ordering allows an algorithm for calculating $U(\theta)$ which saves a considerable amount of time.

The algorithm eliminates the storage problem. Furthermore, we feel the algorithm obtains the estimates quickly enough to make them more attractive. The time for an evaluation of $U(\theta)$ is linear in sample size but the number of iterations decreases due to the asymptotic linearity of $U(\theta)$ (see Jureckova [4]). For example, in a test run with $m = 1000$ and $n = 1200$ the computation time was 0.19 CPU seconds to obtain both the point estimate and confidence interval on an IBM 370 Model 168 (this costs about $0.02).

REFERENCES

1. Antille, A. A linearized version of the Hodges-Lehmann estimator. *Annals of Statistics 2* (Nov. 1974), 1308–1313.
2. Dowell, M., and Jarratt, P. A modified regula falsi method for computing the root of an equation. *BIT 11*, 168–174.
3. Hettmansperger, T.P., and McKean, J.W. A graphical representation for non-parametric inference. *Amer. Statistician 28* (Aug. 1974), 100–102.
4. Jureckova, J. Asymptotic linearity of a rank statistic in regression parameter. *Annals of Math. Statist. 40* (Dec. 1969), 1889–1900.
5. Noether, G.E. *Introduction to Statistics*. Houghton Mifflin, Boston, 1971.

ALGORITHM

```
      SUBROUTINE RANKCI(X, M, Y, N, PERC, DPOINT, DLOW, DHIGH,          RAN   1Ø
     * IERR, ILOW, IHIGH)                                               RAN   2Ø
C  SUBROUTINE TO FIND THE POINT ESTIMATE AND CONFIDENCE INTERVAL         RAN   3Ø
C  RELATED TO THE TWO-SAMPLE RANK TEST (MANN-WHITNEY, WILCOXON),         RAN   4Ø
C  FOR THE PARAMETER D = MU(Y) - MU(X), WHERE MU IS A MEASURE OF         RAN   5Ø
C  LOCATION.                                                            RAN   6Ø
C  THE STATISTICAL ASSUMPTIONS ARE THAT X AND Y ARE RANDOM SAMPLES       RAN   7Ø
C  TAKEN INDEPENDENTLY FROM TWO DIFFERENT POPULATIONS, AND THAT         RAN   8Ø
C  THE POPULATIONS HAVE THE SAME DISTRIBUTION EXCEPT FOR LOCATION.      RAN   9Ø
C  IN PARTICULAR, IT IS ASSUMED THAT THE SCALE PARAMETERS (VARIANCES,   RAN  1ØØ
C  IF THEY EXIST) ARE EQUAL.                                           RAN  11Ø
C  THIS ROUTINE FINDS THE POINT ESTIMATE FOR D AS THAT VALUE DPOINT     RAN  12Ø
C  SUCH THAT THE MANN-WHITNEY U STATISTIC ACHIEVES ITS EXPECTED        RAN  13Ø
C  VALUE FOR THE TEST OF THE NULL HYPOTHESIS  D = DPOINT (THE MIDDLE   RAN  14Ø
C  SUCH VALUE IF IT IS NOT UNIQUE).                                    RAN  15Ø
C  THE CONFIDENCE INTERVAL IS FOUND AS THOSE VALUES OF DD SUCH THAT     RAN  16Ø
C  THE NULL HYPOTHESIS D = DD IS NOT REJECTED BY THE TWO SAMPLE RANK    RAN  17Ø
C  TEST.                                                               RAN  18Ø
C  THE METHOD OF CALCULATING THE TEST STATISTIC U IS THE ORIGINAL       RAN  19Ø
C  METHOD OF MANN AND WHITNEY, MODIFIED TO TAKE ADVANTAGE OF HAVING    RAN  2ØØ
C  THE DATA IN ORDER.  THIS SPEEDS CALCULATIONS CONSIDERABLY, WHICH    RAN  21Ø
C  IS IMPORTANT SINCE THE METHODS USED MUST EVALUATE THE STATISTIC      RAN  22Ø
C  REPEATEDLY.                                                         RAN  23Ø
C  THE POINT AND CONFIDENCE INTERVALS ARE FOUND BY ITERATION, USING    RAN  24Ø
C  TRIMMED MEANS AS STARTING POINTS.  THE BASIC ITERATION PROCEDURE    RAN  25Ø
C  IS A MODIFICATION OF THE REGULA FALSI (LINEAR INTERPOLATION)        RAN  26Ø
C  METHOD, (DOWELL + JARRATT, 1971,BIT) WHICH CONVERGES QUICKLY        RAN  27Ø
C  BECAUSE OF THE ASYMTOTIC LINEARITY OF THE U STATISTIC AS A          RAN  28Ø
C  FUNCTION OF D (JURECKOVA, 1971, ANN. MATH. STAT.).                  RAN  29Ø
C  THE PERCENTAGE OF THE CONFIDENCE INTERVAL IS OBTAINED BY A NORMAL   RAN  3ØØ
C  APPROXIMATION, USING CONTINUITY CORRECTION.                         RAN  31Ø
C  INPUT IS AS FOLLOWS                                                 RAN  32Ø
C       X - ARRAY OF DATA OF DIMENSION M IN NON-DECREASING ORDER       RAN  33Ø
C       M - NUMBER OF OBSERVATIONS IN X SAMPLE (AT LEAST 2)            RAN  34Ø
C       Y - ARRAY OF DATA OF DIMENSION N IN NON-DECREASING ORDER       RAN  35Ø
```

```
C       N - NUMBER OF OBSERVATIONS IN Y SAMPLE (AT LEAST 2)                RAN  36Ø
C       (X,M,Y,AND N ARE UNCHANGED)                                        RAN  37Ø
C       PERC - DESIRED CONFIDENCE PERCENT (BETWEEN 49.999 AND 99.999)      RAN  38Ø
C       (PERC IS CHANGED TO THE NEAREST ATTAINABLE CONFIDENCE)             RAN  39Ø
C  OUTPUT IS AS FOLLOWS                                                    RAN  4ØØ
C       IERR -   Ø IF NORMAL COMPLETION                                    RAN  41Ø
C                1 IF TOO LITTLE DATA (M OR N LESS THAN 2)                 RAN  42Ø
C                2 IF INVALID PERCENTAGE                                   RAN  43Ø
C                3 IF DATA IS NOT IN ORDER                                 RAN  44Ø
C       PERC -   ACTUAL (APPROX.) CONFIDENCE OF THE INTERVAL               RAN  45Ø
C       DPOINT - POINT ESTIMATE OF D                                       RAN  46Ø
C       DLOW -   LOWER CONFIDENCE LIMIT FOR D                              RAN  47Ø
C       ILOW -   THE ORDER OF THE DIFFERENCE DLOW                          RAN  48Ø
C       DHIGH -  UPPER CONFIDENCE LIMIT FOR D                              RAN  49Ø
C       IHIGH -  THE ORDER OF THE DIFFERENCE DHIGH                         RAN  5ØØ
C  WRITTEN JUNE 1975 BY T. RYAN AND J. MCKEAN                              RAN  51Ø
      DIMENSION X(M), Y(N)                                                 RAN  52Ø
C                          INITIALIZE                                      RAN  53Ø
      DPOINT = Ø.Ø                                                         RAN  54Ø
      DLOW = Ø.Ø                                                           RAN  55Ø
      DHIGH = Ø.Ø                                                          RAN  56Ø
      IERR = Ø                                                             RAN  57Ø
      XM = M                                                               RAN  58Ø
      XN = N                                                               RAN  59Ø
C                          ERROR CHECKING                                  RAN  6ØØ
      CALL CHECK(X, M, Y, N, PERC, IERR)                                   RAN  61Ø
      IF (IERR.GT.Ø) RETURN                                                RAN  62Ø
C ***********************************************************************RAN  63Ø
C                          PRELIMINARY ESTIMATE                            RAN  64Ø
      CALL TMEAN(X, M, XBAR, VARXB)                                        RAN  65Ø
      CALL TMEAN(Y, N, YBAR, VARYB)                                        RAN  66Ø
      D = YBAR - XBAR                                                      RAN  67Ø
      V = VARXB + VARYB                                                    RAN  68Ø
      SD = SQRT(V)                                                         RAN  69Ø
      SD = AMAX1(SD,1.ØE-2Ø)                                               RAN  7ØØ
      T = 2.Ø                                                              RAN  71Ø
      IF (M.LT.1Ø .OR. N.LT.1Ø .OR. PERC.GT.96.Ø) T = 3.Ø                  RAN  72Ø
      DL = D - T*SD                                                        RAN  73Ø
      DH = D + T*SD                                                        RAN  74Ø
C ***********************************************************************RAN  75Ø
C                          DEFINE TOLERANCES EPS1 AND EPS2. THESE          RAN  76Ø
C                          TOLERANCES WERE SELECTED FOR IBM 36Ø-37Ø         RAN  77Ø
C                          ACCURACY, APPROXIMATELY 6.5 DECIMAL PLACES.     RAN  78Ø
C                          DEPENDING ON MACHINE BEING USED AND WHETHER     RAN  79Ø
C                          DATA WARRANTS SUCH PRECISION, THE TOLERANCE     RAN  8ØØ
C                          MAY BE LOWERED.                                 RAN  81Ø
C                          BASE TOLERANCE ON RANGE OF DATA                 RAN  82Ø
      Z = X(M) - X(1)                                                      RAN  83Ø
      ZZ = Y(N) - Y(1)                                                     RAN  84Ø
      EPS1 = (1.ØE-7)*AMAX1(Z,ZZ)                                          RAN  85Ø
C                          BASE TOLERANCE ON THE MAGNITUDE OF THE DATA     RAN  86Ø
      K = M/2                                                              RAN  87Ø
      Z = X(K)                                                             RAN  88Ø
      K = N/2                                                              RAN  89Ø
      ZZ = Y(K)                                                            RAN  9ØØ
      EPS2 = (1.ØE-6)*AMAX1(Z,-Z,ZZ,-ZZ)                                   RAN  91Ø
      EPS1 = AMAX1(EPS1,EPS2)                                              RAN  92Ø
C                          BASE TOLERANCE ON WIDTH OF CONFIDENCE INTERVAL  RAN  93Ø
C                          (BASED ON PRELIMINARY ESTIMATES).  THE POINT    RAN  94Ø
C                          ESTIMATE SHOULD BE ACCURATE TO 4 SIG. DIGITS IN RAN  95Ø
C                          THE WIDTH (APPROX.).                            RAN  96Ø
      EPS2 = SD*4.ØE-4                                                     RAN  97Ø
      EPS1 = AMAX1(EPS1,EPS2)                                              RAN  98Ø
C                          MAXIMUM ACCURACY SET TO 1.E-2Ø                  RAN  99Ø
      EPS1 = AMAX1(EPS1,1.ØE-2Ø)                                           RAN 1ØØØ
C                          ONE LESS SIGNIFICANT DIGIT IN END POINTS OF     RAN 1Ø1Ø
C                          THE CONFIDENCE INTERVAL THAN THE POINT ESTIMATE.RAN 1Ø2Ø
      EPS2 = 1Ø.Ø*EPS1                                                     RAN 1Ø3Ø
C ***********************************************************************RAN 1Ø4Ø
C                          GET CRITICAL VALUES OF THE U STATISTIC          RAN 1Ø5Ø
C                          ODD OR EVEN M*N DETERMINES HOW POINT ESTIMATE   RAN 1Ø6Ø
C                          IS CALCULATED                                   RAN 1Ø7Ø
      IODD = 1                                                             RAN 1Ø8Ø
      IF (MOD(M*N,2).EQ.Ø) GO TO 1Ø                                        RAN 1Ø9Ø
      UMED1 = XM*XN/2.Ø                                                    RAN 11ØØ
      GO TO 2Ø                                                             RAN 111Ø
```

```
   1Ø UMED1 = XM*XN/2.Ø - Ø.5                                          RAN 112Ø
      UMED2 = UMED1 + 1.Ø                                              RAN 113Ø
      IODD = Ø                                                         RAN 114Ø
C                       P IS ONE-TAILED PROB.                          RAN 115Ø
   2Ø P = (1ØØ.Ø-PERC)/2ØØ.Ø                                           RAN 116Ø
C                       FIND MEAN AND VARIANCE OF THE U-STATISTIC      RAN 117Ø
C                       FOR USE IN A NORMAL APPROXIMATION.             RAN 118Ø
      UMU = XM*XN/2.Ø                                                  RAN 119Ø
      UVAR = XM*XN*(XM+XN+1.Ø)/12.Ø                                    RAN 12ØØ
      USIGMA = SQRT(UVAR)                                              RAN 121Ø
C                       NORMAL APPROX. (WITH CONTINUITY CORRECTION) IS RAN 122Ø
C                       P = PHI ( (ULOW +.5 -UMU)/USIGMA )             RAN 123Ø
C                       (WHERE PHI IS STANDARD NORMAL DIST. FUNCTION,  RAN 124Ø
C                       AND PHINV IS ITS INVERSE).                     RAN 125Ø
      ULOW = USIGMA*PHINV(P) + UMU - Ø.5                               RAN 126Ø
C                       ROUND CRITICAL VALUE DOWN TO INTEGER           RAN 127Ø
C                       (THIS GIVES CONSERVATIVE BOUNDS), AND FIND P.  RAN 128Ø
      IU = ULOW                                                        RAN 129Ø
      IF (IU.LT.Ø) IU = Ø                                              RAN 13ØØ
      ULOW = IU                                                        RAN 131Ø
      Z = (ULOW+Ø.5-UMU)/USIGMA                                        RAN 132Ø
      P = PHI(Z).                                                      RAN 133Ø
      PERC = 1ØØ.Ø*(1.Ø-2.Ø*P)                                         RAN 134Ø
C                       WANT TO INVERT FUNCTION U AT A HALF INTEGER    RAN 135Ø
      ULOW = ULOW + Ø.5                                                RAN 136Ø
C                       UHIGH                                          RAN 137Ø
      UHIGH = XM*XN - ULOW                                             RAN 138Ø
      ILOW = ULOW + .5                                                 RAN 139Ø
      IHIGH = UHIGH + .5                                               RAN 14ØØ
C                      ILOW  AND IHIGH  GIVE THE ORDERED DIFFERENCES   RAN 141Ø
C                      WHICH FORM THE CONFIDENCE INTERVAL.             RAN 142Ø
C                      AN ESTIMATE OF THE SLOPE OF THE LINEAR          RAN 143Ø
C                      APPROXIMATION TO FMANN.                         RAN 144Ø
      SLOPE = (ABS(Z)*SQRT(XM*XN*(XM+XN)))/((DH-DL)*SQRT(3.))          RAN 145Ø
C                      X1 AND X2 BRACKET THE LOWER CONFIDENCE LIMIT.   RAN 146Ø
C                      THEN BY CALLING THE ROUTINE ILL THE LOWER CONFI- RAN 147Ø
C                      DENCE LIMIT WITHIN EPS2 IS RETURNED VIA DLOW.   RAN 148Ø
      CALL BRACK(DL, ULOW, SLOPE, X, M, Y, N, X1, FX1, X2, FX2)        RAN 149Ø
      CALL ILL(DLOW, ULOW, X1, FX1, X2, FX2, X, M, Y, N, EPS2)         RAN 15ØØ
C                      SAME FOR UPPER END VIA DHIGH.                   RAN 151Ø
      CALL BRACK(DH, UHIGH, SLOPE, X, M, Y, N, X1, FX1, X2, FX2)       RAN 152Ø
      CALL ILL(DHIGH, UHIGH, X1, FX1, X2, FX2, X, M, Y, N, EPS2)       RAN 153Ø
C                      A NEW ESTIMATE OF SLOPE BASED ON THE CONFIDENCE RAN 154Ø
C                      INTERVAL (DLOW,DHIGH), UNLESS THE LENGTH OF     RAN 155Ø
C                      THE INTERVAL IS SMALLER THAN EPS2.              RAN 156Ø
      IF (DHIGH.GT.DLOW+EPS2) SLOPE = ((DH-DL)/(DHIGH-DLOW))*SLOPE     RAN 157Ø
C                      THE SAME ROUTINES ARE USED FOR THE ESTIMATE     RAN 158Ø
C                      DPOINT EXCEPT EPS1 IS USED. THE MIDPOINT        RAN 159Ø
C                      OF THE CONFIDENCE INTERVAL WILL BE THE INITIAL  RAN 16ØØ
C                      ESTIMATE OF DPOINT.                             RAN 161Ø
      D = (DLOW+DHIGH)/2.Ø                                             RAN 162Ø
      CALL BRACK(D, UMED1, SLOPE, X, M, Y, N, X1, FX1, X2, FX2)        RAN 163Ø
      CALL ILL(DPOINT, UMED1, X1, FX1, X2, FX2, X, M, Y, N, EPS1)      RAN 164Ø
C                      IF M*N IS ODD THE ESTIMATE IS DPOINT,           RAN 165Ø
      IF (IODD.EQ.1) RETURN                                            RAN 166Ø
C                      IF EVEN THEN THE VALUE DPOINT IS THE LOWER      RAN 167Ø
C                      CENTER ESTIMATE. THE UPPER CENTER ESTIMATE IS D2,RAN 168Ø
C                      AND THE FINAL ESTIMATE IS THE AVERAGE OF THE TWO.RAN 169Ø
      CALL BRACK(DPOINT, UMED2, SLOPE, X, M, Y, N, X1, FX1, X2, FX2)   RAN 17ØØ
      CALL ILL(D2, UMED2, X1, FX1, X2, FX2, X, M, Y, N, EPS1)          RAN 171Ø
      DPOINT = (DPOINT+D2)/2.                                          RAN 172Ø
      RETURN                                                           RAN 173Ø
      END                                                              RAN 174Ø

      SUBROUTINE CHECK(X, M, Y, N, PERC, IERR)                         CHE   1Ø
C  SUBROUTINE TO DO ERROR CHECKING FOR RANKCI                          CHE   2Ø
C  WRITTEN JUNE 1975 BY T. RYAN                                        CHE   3Ø
      DIMENSION X(M), Y(N)                                             CHE   4Ø
C                       CHECK FOR INSUFFICIENT DATA                    CHE   5Ø
      IF (M.GE.2 .AND. N.GE.2) GO TO 1Ø                                CHE   6Ø
      IERR = 1                                                         CHE   7Ø
      RETURN                                                           CHE   8Ø
C                       CHECK FOR PROPER PERCENT CONFIDENCE            CHE   9Ø
   1Ø IF (PERC.LT.99.999 .AND. PERC.GT.49.999) GO TO 2Ø                CHE  1ØØ
      IERR = 2                                                         CHE  11Ø
      RETURN                                                           CHE  12Ø
```

```
C                       CHECK FOR X AND Y ARRAYS IN NON-DECREASING ORDERCHE  13Ø
   2Ø DO 3Ø I=2,M                                                        CHE  14Ø
        IF (X(I-1).GT.X(I)) GO TO 5Ø                                     CHE  15Ø
   3Ø CONTINUE                                                           CHE  16Ø
      DO 4Ø I=2,N                                                        CHE  17Ø
        IF (Y(I-1).GT.Y(I)) GO TO 5Ø                                     CHE  18Ø
   4Ø CONTINUE                                                           CHE  19Ø
      RETURN                                                             CHE  2ØØ
C                       X OR Y ARRAY OUT OF ORDER                        CHE  21Ø
   5Ø IERR = 3                                                           CHE  22Ø
      RETURN                                                             CHE  23Ø
      END                                                                CHE  24Ø

      SUBROUTINE TMEAN(Z, N, ZBAR, VARZB)                                TME   1Ø
C  GIVEN DATA Z(1), Z(2), ..., Z(N) IN NON-DECREASING ORDER,             TME   2Ø
C  THIS ROUTINE FINDS THE ALPHA-TRIMMED MEAN ZBAR, AND THE               TME   3Ø
C  ESTIMATED VARIANCE OF ZBAR, VARZB.                                    TME   4Ø
C  WRITTEN JUNE 1975 BY T. RYAN                                          TME   5Ø
      DIMENSION Z(N)                                                     TME   6Ø
      DATA ALPHA /Ø.1Ø/                                                  TME   7Ø
      XN = N                                                             TME   8Ø
C                       NT IS NUMBER TRIMMED FROM EACH END               TME   9Ø
C                       N1 IS NUMBER OF LOWEST OBSERVATION NOT TRIMMED   TME  1ØØ
C                       N2 IS NUMBER OF HIGHEST OBSERVATION NOT TRIMMED  TME  11Ø
      NT = ALPHA*XN                                                      TME  12Ø
      N1 = NT + 1                                                        TME  13Ø
      N2 = N - NT                                                        TME  14Ø
C                       TRIMMED MEAN                                     TME  15Ø
      SUM = Ø.Ø                                                          TME  16Ø
      DO 1Ø I=N1,N2                                                      TME  17Ø
        SUM = SUM + Z(I)                                                 TME  18Ø
   1Ø CONTINUE                                                           TME  19Ø
      X = N - 2*NT                                                       TME  2ØØ
      ZBAR = SUM/X                                                       TME  21Ø
C                       WINSORIZED SUM OF SQUARES                        TME  22Ø
      SUM = Ø.Ø                                                          TME  23Ø
      DO 2Ø I=N1,N2                                                      TME  24Ø
        SUM = SUM + (Z(I)-ZBAR)**2                                       TME  25Ø
   2Ø CONTINUE                                                           TME  26Ø
      IF (NT.EQ.Ø) GO TO 3Ø                                              TME  27Ø
      XNT = NT                                                           TME  28Ø
      SUM = SUM + XNT*(Z(N1-1)-ZBAR)**2 + XNT*(Z(N2+1)-ZBAR)**2          TME  29Ø
   3Ø VARZB = SUM/(XN*XN)                                                TME  3ØØ
      RETURN                                                             TME  31Ø
      END                                                                TME  32Ø

      SUBROUTINE ILL(XVAL, FVAL, X1, F1, X2, F2, X, M, Y, N, EPS)        ILL   1Ø
C  THIS ROUTINE SOLVES THE EQUATION                                      ILL   2Ø
C                          F(T)-FVAL=Ø                                   ILL   3Ø
C  FOR A MONOTONE FUNCTION F, IN THIS INSTANCE FMANN. THE METHOD         ILL   4Ø
C  USED IS THE ILLINOIS METHOD AS DESCRIBED BY DOWELL AND JARRATT        ILL   5Ø
C  (1971,BIT), EXCEPT FOR A MODIFICATION WHEN CLOSE TO THE ROOT.         ILL   6Ø
C  THEN IF THE ROOT WAS NOT TRAPPED ON THE LAST ITERATION THE ROU-        ILL   7Ø
C  TINE SWITCHES TO THE BISECTION METHOD.                                ILL   8Ø
C  INPUT -                                                               ILL   9Ø
C           FVAL = THE VALUE OF THE FUNCTION AT THE ROOT.                ILL  1ØØ
C           X1 AND X2 ARE VALUES WHICH BRACKET THE ROOT, THAT IS         ILL  11Ø
C           EITHER                                                       ILL  12Ø
C                        F(X1) .LT. FVAL .LT. F(X2)                      ILL  13Ø
C           OR                                                           ILL  14Ø
C                        F(X2) .LT. FVAL .LT. F(X1)                      ILL  15Ø
C           FX1 = F(X1)                                                  ILL  16Ø
C           FX2 = F(X2)                                                  ILL  17Ø
C           X, M, Y, AND N ARE QUANTITIES USED BY THE FUNCTION FMANN     ILL  18Ø
C           EPS = THE ACCURACY OF THE SOLUTION.                          ILL  19Ø
C  X1, X2, AND THEIR FUNCTIONAL VALUES ARE CHANGED THROUGHOUT THE        ILL  2ØØ
C  ROUTINE.                                                              ILL  21Ø
C  OUTPUT -                                                              ILL  22Ø
C          XVAL = THE ROOT WITHIN EPS.                                   ILL  23Ø
      DIMENSION X(M), Y(N)                                               ILL  24Ø
      F1 = F1 - FVAL                                                     ILL  25Ø
      F2 = F2 - FVAL                                                     ILL  26Ø
      IBISEC = Ø                                                         ILL  27Ø
   1Ø CONTINUE                                                           ILL  28Ø
      IF (ABS(X2-X1).LT.EPS) GO TO 4Ø                                    ILL  29Ø
```

```
C                     X3 IS THE INTERSECTION OF THE SECANT LINE        ILL  3ØØ
C                     FORMED BY (X1,F1), (X2,F2) AND THE X-AXIS.       ILL  31Ø
      X3 = X2 - (F2*(X2-X1))/(F2-F1)                                   ILL  32Ø
      IF (IBISEC.EQ.1) X3 = (X1+X2)/2.                                 ILL  33Ø
      IBISEC = Ø                                                       ILL  34Ø
      CALL FMANN(X3, X, M, Y, N, F3)                                   ILL  35Ø
      F3 = F3 - FVAL                                                   ILL  36Ø
      IF (F3*F2) 2Ø, 2Ø, 3Ø                                            ILL  37Ø
C                     ROOT WAS TRAPPED, SO USE REGULA FALSI            ILL  38Ø
   2Ø X1 = X2                                                          ILL  39Ø
      F1 = F2                                                          ILL  4ØØ
      X2 = X3                                                          ILL  41Ø
      F2 = F3                                                          ILL  42Ø
      GO TO 1Ø                                                         ILL  43Ø
C                     ROOT WAS NOT TRAPPED, SO USE ILLINOIS MOD-       ILL  44Ø
C                     IFICATION.                                       ILL  45Ø
   3Ø X2 = X3                                                          ILL  46Ø
      F2 = F3                                                          ILL  47Ø
      F1 = F1/2.                                                       ILL  48Ø
      IF (ABS(F2).LE.ABS(F1)) GO TO 1Ø                                 ILL  49Ø
C                     IF ILLINOIS MODIFICATION IS MORE RADICAL         ILL  5ØØ
C                     THAN BISECTION, THEN USE BISECTION.              ILL  51Ø
      F1 = 2.Ø*F1                                                      ILL  52Ø
      IBISEC = 1                                                       ILL  53Ø
      GO TO 1Ø                                                         ILL  54Ø
   4Ø XVAL = (X1+X2)/2.                                                ILL  55Ø
      RETURN                                                           ILL  56Ø
      END                                                              ILL  57Ø

      SUBROUTINE BRACK(XINIT, FVAL, SLOPE, X, M, Y, N, X1, FX1, X2,    BRA   1Ø
     * FX2)                                                            BRA   2Ø
C  SUPPOSE THE FUNCTION F IS MONOTONE AND IT IS DESIRED TO SOLVE       BRA   3Ø
C  THE EQUATION                                                        BRA   4Ø
C                    F(T) - FVAL = Ø.                                  BRA   5Ø
C  THIS ROUTINE RETURNS TWO VALUES WHICH BRACKET THE ROOT.             BRA   6Ø
C  INPUT -                                                           BRA   7Ø
C             XINIT = INITIAL ESTIMATE OF THE ROOT.                  BRA   8Ø
C             FVAL = THE VALUE OF THE FUNCTION AT THE ROOT.          BRA   9Ø
C             SLOPE = APPROXIMATE SLOPE OF THE FUNCTION IN A         BRA  1ØØ
C                     NEIGHBORHOOD OF THE ROOT.                      BRA  11Ø
C             X, M, Y, AND N ARE QUANTITIES USED BY THE FUNCTION WHICH BRA 12Ø
C                          IN THIS INSTANCE IS FMANN.                BRA  13Ø
C  NONE OF THE ABOVE QUANTITIES IS CHANGED THROUGHOUT THIS ROUTINE     BRA  14Ø
C  OUTPUT -                                                          BRA  15Ø
C             X1 AND X2 BRACKET THE ROOT, THAT IS EITHER             BRA  16Ø
C                        F(X1) .LT. FVAL .LT. F(X2)                  BRA  17Ø
C             OR                                                     BRA  18Ø
C                        F(X2) .LT. FVAL .LT. F(X1)                  BRA  19Ø
C            FX1 = F(X1)                                             BRA  2ØØ
C             FX2 = F(X2)                                            BRA  21Ø
      DIMENSION X(M), Y(N)                                           BRA  22Ø
      X1 = XINIT                                                     BRA  23Ø
      CALL FMANN(X1, X, M, Y, N, FX1)                                BRA  24Ø
      DELTA = 1.5*((FVAL-FX1)/SLOPE)                                 BRA  25Ø
   1Ø X2 = X1 + DELTA                                                BRA  26Ø
      CALL FMANN(X2, X, M, Y, N, FX2)                                BRA  27Ø
      IF ((FX1-FVAL)*(FX2-FVAL).LT.Ø.) RETURN                        BRA  28Ø
      X1 = X2                                                        BRA  29Ø
      GO TO 1Ø                                                       BRA  3ØØ
      END                                                            BRA  31Ø

      SUBROUTINE FMANN(D, X, MM, Y, NN, U)                           FMA   1Ø
C  ROUTINE TO CALCULATE THE MANN-WHITNEY STATISTIC U.                 FMA   2Ø
C  U DIFFERS ONLY BY A CONSTANT FROM THE WILCOXON 2-SAMPLE RANK       FMA   3Ø
C  STATISTIC W.                                                       FMA   4Ø
C  INPUT -                                                           FMA   5Ø
C            D = NULL HYPOTHESIS VALUE OF THE SHIFT OF THE Y         FMA   6Ø
C                POPULATION FROM THE X POPULATION (I.E. THE NULL     FMA   7Ø
C                HYPOTHESIS BEING TESTED IS  MU(Y) = MU(X) + D).     FMA   8Ø
C            X = ARRAY CONTAINING THE SAMPLE FROM ONE POPULATION,    FMA   9Ø
C                WHICH MUST BE IN NON-DECREASING ORDER.              FMA  1ØØ
C            M = DIMENSION (SAMPLE SIZE) OF X                        FMA  11Ø
C            Y = ARRAY CONTAINING THE SAMPLE FROM THE OTHER POPULATION, FMA 12Ø
C                WHICH MUST BE IN NON-DECREASING ORDER.              FMA  13Ø
C            N = DIMENSION (SAMPLE SIZE) OF Y                        FMA  14Ø
```

```
C  NONE OF THE ABOVE IS ALTERED BY THIS ROUTINE.                          FMA  15Ø
C  OUTPUT -                                                               FMA  16Ø
C            U = MANN-WHITNEY TEST STATISTIC.                            FMA  17Ø
C  THIS ROUTINE IS INTENDED TO BE USED IN AN ITERATION ROUTINE FOR        FMA  18Ø
C  ESTIMATION ONLY, SINCE NO CHECKS OR ADJUSTMENTS ARE MADE FOR TIES.     FMA  19Ø
C  THIS ROUTINE IS DESIGNED TO BE VERY FAST, SINCE IT IS TO BE USED       FMA  2ØØ
C  IN AN ITERATION PROCEDURE, SO NO CHECKS ARE MADE TO INSURE THAT        FMA  21Ø
C  M AND N ARE POSITIVE, OR THAT X AND Y ARE NON-DECREASING.              FMA  22Ø
C  ALL CHECKING SHOULD BE DONE IN CALLING PROGRAM.                        FMA  23Ø
C  U= SUM (NO. OF Y(J) LESS THAN (OR EQUAL) TO X(I)+DELTA) WHERE          FMA  24Ø
C  THE SUM IS OVER I.                                                     FMA  25Ø
C  SINCE ARRAYS X AND Y ARE ORDERED, IF X(I) IS GREATER THAN Y(J),        FMA  26Ø
C  X(I+1) MUST ALSO BE.                                                   FMA  27Ø
C  WRITTEN 3/75 BY T. RYAN.  LAST UPDATED 6/75 BY T. RYAN                 FMA  28Ø
      DIMENSION X(MM), Y(NN)                                             FMA  29Ø
      DELTA = D                                                          FMA  3ØØ
      M = MM                                                             FMA  31Ø
      N = NN                                                             FMA  32Ø
C                       JLE IS THE NUMBER OF Y VALUES LESS THAN OR       FMA  33Ø
C                       EQUAL TO X(I).                                   FMA  34Ø
C                       IU IS ACCUMULATED AS THE VALUE OF THE U          FMA  35Ø
C                       STATISTIC.                                       FMA  36Ø
      JLE = Ø                                                            FMA  37Ø
      IU = Ø                                                             FMA  38Ø
C                       MAIN LOOP                                        FMA  39Ø
      DO 3Ø I=1,M                                                        FMA  4ØØ
        XI = X(I) + DELTA                                                FMA  41Ø
   1Ø   IF (XI.LT.Y(JLE+1)) GO TO 2Ø                                     FMA  42Ø
        JLE = JLE + 1                                                    FMA  43Ø
        IF (JLE.GE.N) GO TO 5Ø                                           FMA  44Ø
        GO TO 1Ø                                                         FMA  45Ø
   2Ø   IU = IU + JLE                                                    FMA  46Ø
   3Ø CONTINUE                                                           FMA  47Ø
   4Ø U = IU                                                             FMA  48Ø
      RETURN                                                             FMA  49Ø
C                       X(I) IS GREATER THAN (OR EQUAL) TO ALL Y(J).     FMA  5ØØ
C                       THEREFORE X(I+1), ..., X(M) ARE ALL GREATER      FMA  51Ø
C                       THAN ALL Y(J).                                   FMA  52Ø
   5Ø IU = IU + (M-I+1)*N                                                FMA  53Ø
      GO TO 4Ø                                                           FMA  54Ø
      END                                                                FMA  55Ø

      FUNCTION PHINV(P)                                                  PHI   1Ø
C  INVERSE NORMAL DISTRIBUTION FUNCTION.                                  PHI   2Ø
C  IF Z IS N(Ø,1), THIS FUNCTION RETURNS PHINV DEFINED BY                 PHI   3Ø
C  P(A .LT. PHINV) = P,        (Ø.Ø  .LT. P AND P .LT. 1.Ø).              PHI   4Ø
C  REF.  HANDBOOK OF MATHEMATICAL FUNCTIONS , 1964, USDC, NATIONAL        PHI   5Ø
C  BUREAU OF STANDARDS, WASH.  DC. P. 933, FORMULA 26.2.23.               PHI   6Ø
C  ERROR WILL BE LESS THAN 4.5E-Ø4.                                       PHI   7Ø
C  WRITTEN 4/76 BY T. RYAN BASED ON ROUTINE BY H. D. KNOBLE (1966).       PHI   8Ø
      IF (P.LE.Ø. .OR. P.GE.1.Ø) GO TO 3Ø                                PHI   9Ø
      IF (P.LT.Ø.5) GO TO 1Ø                                             PHI  1ØØ
      SIGN = 1.Ø                                                         PHI  11Ø
      PTAIL = 1.Ø - P                                                    PHI  12Ø
      GO TO 2Ø                                                           PHI  13Ø
   1Ø SIGN = -1.Ø                                                        PHI  14Ø
      PTAIL = P                                                          PHI  15Ø
   2Ø T = SQRT(ALOG(1.Ø/(PTAIL*PTAIL)))                                  PHI  16Ø
      Z = ABS(T-(2.515517+T*(Ø.8Ø2853+T*Ø.Ø1Ø328))/(1.Ø+T*               PHI  17Ø
     * (1.432788+T*(Ø.189269+T*Ø.ØØ13Ø8))))                             PHI  18Ø
      PHINV = Z*SIGN                                                     PHI  19Ø
      RETURN                                                             PHI  2ØØ
   3Ø CONTINUE                                                           PHI  21Ø
      RETURN                                                             PHI  22Ø
      END                                                                PHI  23Ø

      FUNCTION PHI(X)                                                    PHI   1Ø
C  NORMAL DISTRIBUTION FUNCTION                                           PHI   2Ø
C  LET Z BE N(Ø,1), THEN PHI IS DEFINED BY PHI = P(Z .LE. X).             PHI   3Ø
C  REF.   HANDBOOK OF MATHEMATICAL FUNCTIONS , 1964, USDC, NATIONAL       PHI   4Ø
C  BUREAU OF STANDARDS, WASH. DC, P. 933, FORMULA 26.2.19.                PHI   5Ø
C  ERROR IS LESS THAN 1.5E-Ø7.                                            PHI   6Ø
C  WRITTEN 4/76 BY T. RYAN, BASED ON ROUTINE BY H. D. KNOBLE (1966).      PHI   7Ø
      Z = X                                                              PHI   8Ø
      IF (Z.LT.Ø.Ø) GO TO 1Ø                                             PHI   9Ø
```

```
      ISIGN = 1                                                          PHI  1ØØ
      GO TO 2Ø                                                           PHI  11Ø
   1Ø Z = -X                                                             PHI  12Ø
      ISIGN = -1                                                         PHI  13Ø
   2Ø P = Ø.5*(1.Ø+Z*(Ø.Ø49867347+Z*(Ø.Ø21141ØØ61+Z*(3.2776263D-3+Z*     PHI  14Ø
     * (3.8ØØ36E-Ø5+Z*(4.889Ø6E-Ø5+Z*5.383E-Ø6))))))**(-16)             PHI  15Ø
      IF (ISIGN.EQ.1) P = 1.Ø - P                                        PHI  16Ø
      PHI = P                                                            PHI  17Ø
      RETURN                                                             PHI  18Ø
      END                                                                PHI  19Ø
```

517-P 1- 0

ALGORITHM 517
A Program for Computing the Condition Numbers of Matrix Eigenvalues Without Computing Eigenvectors [F2]

S. P. CHAN, R. FELDMAN, and B. N. PARLETT
University of California

Key Words and Phrases: eigenvalue, condition number
CR Categories: 5.14
Language: Fortran

DESCRIPTION

1. Theoretical Background

1.1 *The Sensitivity of Eigenvalues.* Several good programs are available for the computation of the eigenvalues of real and complex matrices [2, 3, 7]. Because of the limitations of finite precision arithmetic, these programs cannot produce, in general, the exact eigenvalues of the given matrix A. However the computed numbers are always (very close to) the eigenvalues of a matrix $A + E$ which is very close to A. This matrix E is not unique and error analyses [6] have shown the existence of E's with satisfactorily small upper bounds on $\| E \| / \| A \|$. Here $\| \cdot \|$ denotes an appropriate matrix norm.

It follows from these remarks that a good program will not always deliver accurate approximations to the eigenvalues of A. It can happen that some, or all, of the eigenvalues are very sensitive to changes in the matrix elements; so some, or all, of the eigenvalues of $A + E$ may differ sharply from those of A. Actually this is true only for non-normal matrices. Real symmetric matrices—indeed all normal matrices—determine their eigenvalues very well; the change induced in an eigenvalue of such an A cannot exceed the spectral norm of E (which is defined below).

Two questions arise: How can this sensitivity be measured, and how cheaply can it be computed?

Simple eigenvalues. To any simple eigenvalue λ of A there correspond both a column vector x and a row eigenvector y^* (the conjugate transpose of y) which are unique to within a scalar multiple. Thus

$$Ax = x\lambda, \qquad y^*A = \lambda y^* \tag{1}$$

Received 22 March 1976 and 7 September 1976.

This research was sponsored by the Office of Naval Research under Contract N00014-69-A-0200-1017.
Authors' present addresses: S.P. Chan, Burroughs Corp., Paoli, PA; R. Feldman and B. N. Parlett, Department of Mathematics, University of California, Berkeley, CA 94720.

and $x = y$ if A is normal (i.e. $A^*A = AA^*$). The most popular measure of the sensitivity of λ was called the spectral condition number cond (λ) by Wilkinson [6]. Let θ denote the acute angle between x and y; then

$$\text{cond } (\lambda) \equiv \text{secant } \theta = \| y \| \cdot \| x \| / | y^*x |$$

where

$$\| v \| = \sqrt{(v^*v)}. \tag{2}$$

This definition gives a number in $[1, \infty)$ which increases monotonically with the sensitivity of λ to changes in A.

In order to justify this definition two popular matrix norms will be used:

$$\begin{aligned} \| M \| &\equiv \max_{v \neq 0} \| Mv \| / \| v \| = \sqrt{(\lambda_{\max}(M^*M))}, \\ \| M \|_E &\equiv \sqrt{(\sum_i \sum_j | m_{ij} |^2)} = \sqrt{(\text{trace } (M^*M))}. \end{aligned} \tag{3}$$

Let $| \delta\lambda |$ be the change in λ corresponding to a change δA in A. It can be shown that

$$\text{cond } (\lambda) = \sup | \delta\lambda | / \| \delta A \|_E \tag{4}$$

over all non-null infinitesimal δA.

Another useful characterization of cond (λ) is the following: The spectral projector P_λ of λ is the matrix which projects every vector into a multiple of the eigenvector of λ. It is easy to verify that for simple λ

$$P_\lambda = xy^*/y^*x \tag{5}$$

where y^*x is a scalar, and by using the fact that P_λ is of rank 1, one can show that

$$\text{cond } (\lambda) = \| P_\lambda \|_E = \| P_\lambda \|. \tag{6}$$

It is this characterization which can be generalized.

Multiple eigenvalues. When λ has geometric multiplicity m, almost all perturbations of A break λ into m simple eigenvalues in such a way that $\sup | \delta\lambda | / \| \delta A \|$ is unbounded. Thus it is customary to set cond $(\lambda) = \infty$ in this case.

There is more to be said however. A reasonable definition (see Kahan [4]) puts

$$\text{cond } (\lambda) \equiv \sup | \delta\lambda | / \| \delta A \| \tag{7}$$

over all non-null infinitesimal δA which preserve the multiplicity of λ. This number can be estimated because cond $(\lambda) \leq \| P_\lambda \|_E / m$ where the spectral projector P_λ satisfies $AP_\lambda = P_\lambda A = \lambda P_\lambda + N_\lambda$ and N_λ is nilpotent (i.e. $N_\lambda{}^m = 0$). Moreover P_λ can be found from the expression

$$P_\lambda = X(Y^*X)^{-1}Y^* \tag{8}$$

where the columns of X and rows of Y^* are bases for the invariant subspaces of λ.

We have followed the usual practice (of setting the condition number equal to infinity) in our program CONDIT but wish to point out that it is feasible to bring into adjacent positions on the diagonal of the Schur form any associated ill-conditioned eigenvalues. The spectral projector for this group of eigenvalues can then be found from (8), and if its norm is small, the group can be designated as a cluster. That is another project.

If more specific information is required, then the individual elements of P_λ will be involved because

$$\partial\lambda/\partial a_{ij} = e_j{}^* P_\lambda e_i \tag{9}$$

with λ simple.

A warning should be offered at this point. The measures presented above are based on the Euclidean vector norm and the convention that A acts on vectors in Euclidean n-space. It can happen that this model is quite inappropriate for certain applications and then the conventional condition numbers will be irrelevant. However it is only the order of magnitude (base 10) of cond (λ) which is wanted, in most cases, and this will be constant over a large range of norms.

1.2 *Invariance Properties.* When the role of the matrix is to be stressed the condition number is written cond (λ, A).

THEOREM. *If Q is unitary, i.e. $Q^*Q = QQ^* = I$, and λ is a simple eigenvalue of A, then*

$$cond\ (\lambda, QAQ^*) = cond\ (\lambda, A).$$

PROOF. Let $Ax = x\lambda$, $y^*A = \lambda y^*$. Then

$$(QAQ^*)(Qx) = (Qx)\lambda, \qquad (y^*Q^*)(QAQ^*) = \lambda(y^*Q^*).$$

Because λ is simple, $y^*x \neq 0$ and

$$\begin{aligned} \text{cond}\ (\lambda, QAQ^*) &= \| y^*Q^* \| \cdot \| Qx \| / | (y^*Q^*)(Qx) |, \\ &= \| y^* \| \cdot \| x \| / | y^*x |, \\ &= \text{cond}\ (\lambda, A), \end{aligned}$$

because the Euclidean norm is unitarily invariant. □

COROLLARY. *If a given matrix B is reduced to Hessenberg form H by unitary similarities (such as Householder transformations) and the QR algorithm is applied to H to produce, in the limit, a quasi-triangular matrix T, then*

$$cond\ (\lambda, B) = cond\ (\lambda, T).$$

1.3 *The Use and Cost of Condition Numbers.* A computed eigenvalue λ of a given matrix A is an exact eigenvalue of many matrices including some close to A. Let $A + E$ designate one of the closest matrices. Provided $(\| E \|_E / \| A \|_E)^2$ is negligible, the error in λ is bounded by cond $(\lambda) \| E \|_E$. Error analyses [6] give an upper bound β on $(\| E \|_E / \| A \|_E)$ when the Householder/QR method is used. It follows that $\log_{10} (| \lambda | / \beta \cdot \text{cond}\ (\lambda) \cdot \| A \|_E)$ gives the number of decimal digits in λ which are assuredly correct.

When no figures in λ can be relied on, then a warning tag should be attached to λ for most applications. Conversely, when an adequate number of figures are certified as correct in each eigenvalue of A, then the subsequent calculations are placed on a sounder footing.

These estimates of the number of correct figures have proved useful in the comparison of rival eigenvalue programs and in debugging big programs of which the eigenvalue calculations were merely a part.

A natural question at this stage is how much extra does it cost to compute cond (λ) as well as λ? The answer must depend on whether the user also computes x and/or y along with λ. We focus on real matrices and real arithmetic.

(1) If a complete Jordan factorization $A = X\Lambda Y^*$ $(Y^*X = I)$ is computed then each cond (λ_i) can be found from the definition $\| x_i \| \| y_i^* \| / | y_i^*x |$ at negligible extra cost in storage and time. No special program is needed and this case will not be considered further. Few dependable Jordan factorization routines are currently available.

(2) If a program is used which yields X and Λ but not Y^*, then it is necessary to compute the triangular factorization L_xU_x and store it in an extra array. Then cond $(\lambda_i) = \| e_i^*X^{-1} \| \cdot \| Xe_i \|$. To invert X costs n^3 basic operations whereas X and Λ may be found in approximately $7n^3$ operations using the double QR transformation.

No special program is needed. The time penalty is slight but the extra storage requirement is substantial. This case will not be discussed further.

(3) The eigenvalues λ of A may be found (EISPACK path, ELMHES, HQR) in under $4\frac{1}{6}n^3$ operations with no supplementary $n \times n$ storage arrays provided A can be overwritten. This is the most interesting case. No extra arrays are needed for the computation of cond (λ_i), $i = 1, \ldots, n$, but the multiplication count rises to approximately $7n^3$. See Section 1.4 for more details. The $O(n^2)$ terms bring down the ratio of running times; the increase is approximately 50 percent (± 15 percent).

Our method is easily described. The given matrix A is reduced to Hessenberg form H by *orthogonal* similarity transformations. Then H is transformed to quasi-triangular Schur form T by the double QR algorithm working on the whole of H and not just the remaining principal submatrices. None of the orthogonal transforming matrices is retained. Finally the column and row eigenvectors of T are found, for each λ, by back substitution and then discarded immediately after cond (λ) has been calculated.

By Theorem 1, cond (λ, T) = cond (λ, A).

For simplicity all condition numbers exceeding 10^{30} are recorded as 10^{30}.

The program uses only real arithmetic even if A has complex eigenvalues.

1.4 *Operation Counts.* In [5] Parlett and Wang point out that straightforward counts of multiplications and additions are unreliable indicators of running times. Nevertheless they are good to within a factor of 2 and they do give insight into the way the algorithm spends its time. An op is defined as a scalar multiplication or division followed by an addition.

ORTHES. The $(n - j)$th step transforms the last j rows and columns while reducing column $(n - j)$ to upper Hessenberg form.

Row operations: $A \to A' = A - w\gamma(w^{\mathrm{T}}A)$, $\gamma = 2/w^{\mathrm{T}}w$, $w^{\mathrm{T}} = (0, \ldots, 0, x, \ldots, x)$

Computation	γ	$w^{\mathrm{T}}A$	$v^{\mathrm{T}} = \gamma(w^{\mathrm{T}}A)$	$A - wv^{\mathrm{T}}$	Total
Cost	j	j^2	j	j^2	$2j(j + 1)$

Column operations: $A' \to A'' = A' - \gamma A'ww^{\mathrm{T}}$

Computation	γ	$A'w$	$u = \gamma A'w$	$A' - uw^{\mathrm{T}}$	Total
Cost	0	nj	n	nj	$n(2j +)$

Grand total: $\sum_{j=1}^{n-1} n(2j + 1) + 2j(j + 1) = \frac{4}{3}n(n^2 - 1)$

The program ELMHES is approximately twice as fast as ORTHES but will not preserve condition numbers.

CONDIT. It suffices to assume that all eigenvalues are real. To find the column and row eigenvectors for the jth eigenvalue requires backsolving triangular systems of $(j - 1)$ and $(n - j - 1)$ equations, respectively.

Computation	x	y^*	Cond
Cost	$\sum_{i=1}^{j-1} i$	$\sum_{i=1}^{n-j-1} i$	n

Grand total: $\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$

HQR. A typical double QR transformation acts on a $j \times j$ submatrix of a Hessenberg matrix. To restore column k to Hessenberg form requires the following operations for $k \leq j - 2$. For $k = j - 1$ the cost is $3j + 6$ ops.

Computation	Key quantities	Row operations	Column operations
Cost	9	$\sum_{l=k}^{j} 5$	$\sum_{l=1}^{\min(k+3,j)} 5$

Total: $\sum_{k=1}^{j-2} [9 + 5(j - k + 1) + 5(k + 3)] + 3j + 6 = 5j^2 + 22j - 54$

Assume four initial full transformations with $j = n$ and then two iterations per eigenvalue.

Grand total: $\frac{10}{3}n^3 + 47n^2 + \frac{11}{3}n - 182$

QR2N0Z. The same transformations as in HQR must act on the whole matrix. This changes the range of the row operation and not the column because the $j \times j$ submatrix being transformed is the leading principal submatrix.

Computation	Key quantities	Row operations	Column operations
Cost	9	$\sum_{l=k}^{n} 5$	$\sum_{l=1}^{\min(k+3,j)} 5$

Total: $\sum_{k=1}^{j-2} [9 + 5(n - k + 1) + 5(k + 3)] + 3n + 21 = (5n + 29)(j - 2) + (3n + 21)$

With the same assumptions as above,

Grand total: $5n^3 + 40n^2 + 23n - 300$

A summary of operation counts is:

ELMHES + HQR:	$4\frac{1}{6}n^3 + 47n^2 + 3n - 182$;
ORTHES + HQR:	$5n^3 + 47n^2 + 2n - 182$;
ORTHES + QR2N0Z + CONDIT:	$7n^3 + 40\frac{1}{2}n^2 + 21\frac{1}{2}n - 300$.

The actual timings were more favorable to our program than these operation counts suggest. The assumption of two iterations per eigenvalue is unrealistic. In practice there are more iterations with larger values of j and fewer with small values. With $20 \leq n \leq 60$ our program ran, on the average, 50 percent longer than did ELMHES + HQR; the worst case ran 65 percent longer.

2. Applicability

The program accepts real square matrices which can be stored in the high speed memory of the computer.

The condition numbers of all eigenvalues of all normal matrices (and this includes symmetric matrices) are unity and consequently the program is intended for use with non-normal matrices.

Before our programs QR2N0Z and CONDIT are used, A should be reduced to Hessenberg form H by orthogonal congruences. We recommend the procedure ORTHES described in [7] and its Fortran counterpart ORTHES described in [2, p. 297].

Our program QR2N0Z is an adaptation of HQR2 [2, p. 248] and is designed to avoid the formation of the product of all the similarity transformations used in the double QR algorithm and the calculation of the eigenvectors of the final matrix of the QR sequence.

3. Organizational Details

3.1 *Standardization.* (i) In the course of the QR algorithm applied to H it is possible for two real eigenvalues to be found at the same time as the roots of a 2×2 diagonal block

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}.$$

It is convenient in such cases to do a supplementary plane rotation which will reduce this diagonal block to upper triangular form and change the corresponding rows and columns of H accordingly.

If this transformation is done at the time the eigenvalues λ_1 and λ_2 are recorded then some of the quantities which determine the correct angle of rotation will be available.

This device is employed in HQR2 and has been carried over to QR2N0Z. The details are given below.

The parameters $c = \cos\theta$, $s = \sin\theta$ are determined so that

$$\begin{pmatrix} c & -s \\ s & c \end{pmatrix}\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\begin{pmatrix} c & s \\ -s & c \end{pmatrix}$$

is upper triangular. Thus $\gamma c^2 - dcs - \beta s^2 = 0$, $d = \delta - \alpha$. Let $t = (d/2)^2 + \beta\gamma$; then $\cot\theta = (d/2 + \text{sign}\,(d)\,\sqrt{t})/2\gamma$, $s = \text{sign}\,(\cot\theta)(1 + \cot^2\theta)^{-1/2}$, $c = s\cdot\cot\theta$.

(ii) It is also convenient to perform a supplementary plane rotation after a pair of complex conjugate eigenvalues, $\lambda \pm i\mu$, has been recorded in the course of the QR algorithm. In this case the transformation of the diagonal block is

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\begin{pmatrix} c & -s \\ s & c \end{pmatrix} = \begin{pmatrix} \lambda & \theta \\ \xi & \lambda \end{pmatrix}$$

where $\xi\theta = -\mu^2$. This device is not used in HQR2.

Note that it is not in general possible to transform

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \rightarrow \begin{pmatrix} \lambda & -\mu \\ \mu & \lambda \end{pmatrix}$$

using *orthogonal* similarity transformations.

The purpose of the transformation is to yield a simple solution to certain systems of linear equations which must be solved. The supplementary plane rotation is done at the stage when the eigenvalues are being recorded in QR2N0Z. In this case $t \equiv p^2 + \beta\gamma < 0$, $p = (\alpha - \delta)/2$. We want to choose $c = \cos\theta$ and $s = \sin\theta$ so that

$$\alpha c^2 + (\beta + \gamma)cs + \delta s^2 = \delta c^2 - (\beta + \gamma)cs + \alpha s^2.$$

Hence

$$\tan 2\theta = 2sc/(c^2 - s^2) = -2p/\sigma = (2 \mid p \mid / \mid \sigma \mid) \operatorname{sign}(-p\sigma), \quad \sigma = \beta + \gamma.$$

Let $\tau = \sqrt{(\sigma^2 + 4p^2)}$. Then

$$\cos\theta = q = \sqrt{(\tfrac{1}{2}(1 + \cos 2\theta))} = \sqrt{((1 + \mid \sigma \mid / \tau)/2)},$$

$$\sin\theta = \sin 2\theta/2 \cos\theta = \mid p \mid \operatorname{sign}(-p\sigma)/\tau q.$$

3.2 *The Computation of the Eigenvectors of a Standardized Real, Block Upper Triangular Matrix.* For each real eigenvalue λ the eigenvectors u, w^* satisfy $Tu = u\lambda$, $w^*T = \lambda w^*$.

For each complex conjugate pair of eigenvalues, $\lambda \pm i\mu$, the eigenvectors $u_1 \pm iu_2$ ${w_1}^* \mp i{w_2}^*$ satisfy $T(u_1, u_2) = (u_1, u_2)\hat{E}$, $(w_1, w_2)^*T = \hat{E}(w_1, w_2)^*$, where

$$\hat{E} = \begin{pmatrix} \lambda & \mu \\ -\mu & \lambda \end{pmatrix}.$$

In effecting the back substitution process in real arithmetic there are four different cases which can occur, depending on whether the matrices D and E shown below are 1×1 or 2×2:

$$T = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ & D & \cdot & \cdot & \cdot \\ & & \cdot & \cdot & \cdot \\ & & \mathrm{O} & E & \cdot \\ & & & & \cdot \end{pmatrix} \qquad D = \begin{cases} \alpha \text{ or} \\ \begin{pmatrix} \alpha & \beta \\ \gamma & \alpha \end{pmatrix} \end{cases}$$

$$E = \begin{cases} \lambda \text{ or} \\ \begin{pmatrix} \lambda & \phi \\ \theta & \lambda \end{pmatrix} \end{cases}$$

where $\theta\phi = -\mu^2$. The positions of D and E should be exchanged when considering the row eigenvectors.

Type 1. *pair-pair* (E *is* 2×2, D *is* 2×2). Imagine that the elements of u_1, u_2 in the same row as D are about to be computed. All elements below these have already been found, the elements below E being zero.

Let $j1, j2$ be the rows of T in which D lies. Then the unknowns are

$$V = \begin{pmatrix} u_1(j1) & u_2(j1) \\ u_1(j2) & u_2(j2) \end{pmatrix}.$$

The equation to be solved in the column case is $-DV + V\hat{E} = R$, where

$$R = \begin{pmatrix} r_1(j1) & r_2(j1) \\ r_1(j2) & r_2(j2) \end{pmatrix}, \quad r_\nu(m) = \sum_{k=j2+1}^{i+1} t_{m,k}u_\nu(k), \begin{cases} m = j1, j2, \\ \nu = 1, 2. \end{cases} \tag{10}$$

In the row case let

$$V = \begin{pmatrix} w_1(j1) & w_2(j1) \\ w_1(j2) & w_2(j2) \end{pmatrix};$$

then the equations to be solved are

$$-V^{T}D + \hat{E}V^{T} = R^{T}, \tag{11}$$

where R is as above except that k runs from i to $j1 - 1$. Transposing yields

$$-D^{T}V + V\hat{E}^{T} = R.$$

Comparing this with the column case we see that it is only necessary to transpose D and $\hat{E}$ (i.e. to exchange β, γ and $\mu, -\mu$) in order to use the same code for both cases.

The way in which these four linear equations in four unknowns are solved is described in Section 3.3.

Type 2. *pair-single* (E *is* 2×2, D *is* 1×1). The relevant equations are

$$-\alpha(u_1(j), u_2(j)) + (u_1(j), u_2(j))\hat{E} = (r_1(j), r_2(j))$$

and

$$-\alpha(w_1(j), w_2(j)) + (w_1(j), w_2(j))\hat{E}^{T} = (r_1(j), r_2(j)).$$

Let $d = \lambda - \alpha$, den $= d^2 + \mu^2$, val $= \begin{cases} \mu \text{ (for column)} \\ -\mu \text{ (for row)}. \end{cases}$ The solution for both cases is

$$\begin{aligned} v_1 &= (r_1 \cdot d + r_2 \cdot \text{val})/\text{den}, \\ v_2 &= (-r_1 \cdot \text{val} + r_2 \cdot d)/\text{den}. \end{aligned} \tag{12}$$

Type 3. *single-pair* (E *is* 1×1, D *is* 2×2). The relevant equations are

$$-D\begin{pmatrix} u_1(j1) \\ u_1(j2) \end{pmatrix} + \begin{pmatrix} u_1(j1) \\ u_1(j2) \end{pmatrix}\lambda = \begin{pmatrix} r_1(j1) \\ r_1(j2) \end{pmatrix}$$

and the same equation for w_1 with D^{T} in place of D. Set $d = \lambda - \alpha$, den $= d^2 - \beta\gamma$. The solution is

$$\begin{aligned} v_1 &= (r_1(j1) \cdot d + r_1(j2) \cdot \tilde{\beta})/\text{den}, \\ v_2 &= (r_1(j1) \cdot \tilde{\gamma} + r_1(j2) \cdot d)/\text{den}, \end{aligned} \tag{13}$$

where $\tilde{\beta} = \begin{Bmatrix} \beta & \text{(for column)} \\ \gamma & \text{(for row} \end{Bmatrix}$ and $\tilde{\gamma} = \begin{Bmatrix} \gamma & \text{(for column)} \\ \beta & \text{(for row} \end{Bmatrix}$. In practice $\tilde{\beta} = T(JJ, J), \tilde{\gamma} = T(J, JJ)$ and the indices J and JJ are given the values of $J1$ and $J2$ in the appropriate order.

Type 4. *single-single* (E *is* 1×1, D *is* 1×1). $v_1 = r_1(j)/\text{den}$, den $= \lambda - \alpha$.

Type 5. *formula breakdown.* If in any of the previous cases $D = E$ then the formulas for solution break down. There are two cases to consider.

(i) Linear Independence. Any element v_j for which the formula yields 0/0 can be set to any value, the most convenient is 0. This represents the existence of a whole space of eigenvectors associated with E.

(ii) Defective Case. Any element v_j for which the formula yields a value exceeding 1/TOL will cause the condition number to exceed 1/TOL. If this case is detected computation is interrupted, the condition number is set to 1/TOL, and the program proceeds to the next eigenvalue.

We propose that TOL $= 10^{-30}$ will be suitable for most applications and most computers.

These tests make the code simple and machine independent. However, it is possible to devise matrices for which the given value 1/TOL for the condition is very misleading. We know of no failsafe procedure which does not involve deciding the rank of $T - \xi$ for all ξ in a neighborhood of λ. This is a costly, difficult, and often unrewarding task.

3.3 *Closed Form Solution for Equations of Type* 1. The equations to be solved are of the form $-DV + V\hat{E} = R$, where

$$D = \begin{pmatrix} \alpha & \beta \\ \gamma & \alpha \end{pmatrix}, \quad \hat{E} = \begin{pmatrix} \lambda & \mu \\ -\mu & \lambda \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix}.$$

The standardization of the block triangular matrix T forces the diagonal elements of D to be equal. This yields simple formulas for the elements of V.

Rewrite the equation as

$$(v_{11}\ v_{12}\ v_{21}\ v_{22})\begin{pmatrix} B & -\gamma I_2 \\ -\beta I_2 & B \end{pmatrix} = (r_{11}\ r_{12}\ r_{21}\ r_{22}) = r^{\mathrm{T}},$$

where

$$B = \begin{pmatrix} \lambda - \alpha & \mu \\ -\mu & \lambda - \alpha \end{pmatrix}.$$

Observe that

$$\begin{pmatrix} B & -\gamma I_2 \\ -\beta I_2 & B \end{pmatrix}\begin{pmatrix} B^{\mathrm{T}} & \gamma I_2 \\ \beta I_2 & B^{\mathrm{T}} \end{pmatrix} = \tau I_4 + 2\mu J_4,$$

where $\tau = (\lambda - \alpha)^2 + \mu^2 - \beta\gamma$, with $\beta\gamma < 0$,

$$J_4 = \begin{pmatrix} 0 & 0 & 0 & \gamma \\ 0 & 0 & -\gamma & 0 \\ 0 & \beta & 0 & 0 \\ -\beta & 0 & 0 & 0 \end{pmatrix},$$

and I_k is the $k\times k$ identity matrix. Further, $(\tau I_4 + 2\mu J_4)(\tau I_4 - 2\mu J_4) = [\tau^2 - 4\mu^2(-\beta\gamma)]I_4$. Hence

$$(v_{11}\ v_{12}\ v_{21}\ v_{22})(\tau^2 + 4\mu^2\beta\gamma) = r^{\mathrm{T}}\begin{pmatrix} B^{\mathrm{T}} & \gamma I_2 \\ \beta I_2 & B^{\mathrm{T}} \end{pmatrix}(\tau I_4 - 2\mu J_4)$$

$$= r^{\mathrm{T}}\begin{pmatrix} e & -f & \gamma g & -\gamma h \\ f & e & \gamma h & \gamma g \\ \beta g & -\beta h & e & -f \\ \beta h & \beta g & f & e \end{pmatrix}$$

where $d = \lambda - \alpha$, $e = d\tau$, $f = \mu(\tau + 2\beta\gamma)$, $g = \tau - 2\mu^2$, and $h = 2\,d\mu$. Note that

$$\begin{aligned} \tau^2 + 4\mu^2\beta\gamma &= (d^2 + \mu^2 - \beta\gamma)^2 + 4\mu^2\beta\gamma \\ &= g^2 + h^2 \\ &(= 0 \quad \text{if and only if} \quad \alpha = \lambda,\ \mu^2 = -\beta\gamma). \end{aligned}$$

These same formulas will be valid for the row eigenvectors provided we exchange (β, γ) and $(\mu, -\mu)$.

The alternative to using this closed form solution is to code a special version of Gaussian elimination with pivoting. It is the pivoting which would lengthen the code considerably.

3.4 *The Condition Number of Conjugate Pairs of Eigenvalues.* Let $\lambda \pm i\mu$ be a complex pair of eigenvalues of the real Schur matrix T obtained by the QR algorithm. In the course of the algorithm the following real equations are solved for real n-vectors u_1, u_2, w_1, w_2 :

$$T(u_1, u_2) = (u_1, u_2)\begin{pmatrix} \rho & \mu \\ -\mu & \rho \end{pmatrix}, \quad (w_1, w_2)^* T = \begin{pmatrix} \rho & \mu \\ -\mu & \rho \end{pmatrix}(w_1, w_2)^*. \tag{5}$$

Thus span (u_1, u_2) and span (w_1^*, w_2^*) are real invariant subspaces under T. However, $\{u_1, u_2\}$ and $\{w_1^*, w_2^*\}$ are very special bases of these spaces.

LEMMA. *With the notation given above, $u_1 \pm iu_2$ and $w_1^* \mp iw_2^*$ are the column and row eigenvectors belonging to $\lambda \pm i\mu$.*

PROOF. From (5),

$$\begin{aligned} Tu_1 &= u_1\lambda - u_2\mu, \quad w_1^* T = \lambda w_1^* + \mu w_2^*, \\ Tu_2 &= u_1\mu + u_2\lambda, \quad w_2^* T = -\mu w_1^* + \lambda w_2^*. \end{aligned}$$

Hence

$$T(u_1 + iu_2) = u_1(\lambda + i\mu) + iu_2(i\mu + \lambda),$$

then the equations to be solved are

$$-V^{\mathrm{T}}D + \hat{E}V^{\mathrm{T}} = R^{\mathrm{T}}, \tag{11}$$

where R is as above except that k runs from i to $j1 - 1$. Transposing yields

$$-D^{\mathrm{T}}V + V\hat{E}^{\mathrm{T}} = R.$$

Comparing this with the column case we see that it is only necessary to transpose D and $\hat{E}$ (i.e. to exchange β, γ and μ, $-\mu$) in order to use the same code for both cases.

The way in which these four linear equations in four unknowns are solved is described in Section 3.3.

Type 2. *pair-single* (E *is* 2×2, D *is* 1×1). The relevant equations are

$$-\alpha(u_1(j), u_2(j)) + (u_1(j), u_2(j))\hat{E} = (r_1(j), r_2(j))$$

and

$$-\alpha(w_1(j), w_2(j)) + (w_1(j), w_2(j))\hat{E}^{\mathrm{T}} = (r_1(j), r_2(j)).$$

Let $d = \lambda - \alpha$, den $= d^2 + \mu^2$, val $= \begin{cases} \mu \text{ (for column)} \\ -\mu \text{ (for row)}. \end{cases}$ The solution for both cases is

$$\begin{aligned} v_1 &= (r_1 \cdot d + r_2 \cdot \text{val})/\text{den}, \\ v_2 &= (-r_1 \cdot \text{val} + r_2 \cdot d)/\text{den}. \end{aligned} \tag{12}$$

Type 3. *single-pair* (E *is* 1×1, D *is* 2×2). The relevant equations are

$$-D\begin{pmatrix} u_1(j1) \\ u_1(j2) \end{pmatrix} + \begin{pmatrix} u_1(j1) \\ u_1(j2) \end{pmatrix}\lambda = \begin{pmatrix} r_1(j1) \\ r_1(j2) \end{pmatrix}$$

and the same equation for w_1 with D^{T} in place of D. Set $d = \lambda - \alpha$, den $= d^2 - \beta\gamma$. The solution is

$$\begin{aligned} v_1 &= (r_1(j1) \cdot d + r_1(j2) \cdot \tilde{\beta})/\text{den}, \\ v_2 &= (r_1(j1) \cdot \tilde{\gamma} + r_1(j2) \cdot d)/\text{den}, \end{aligned} \tag{13}$$

where $\tilde{\beta} = \begin{cases} \beta & \text{(for column)} \\ \gamma & \text{(for row} \end{cases}$ and $\tilde{\gamma} = \begin{cases} \gamma & \text{(for column)} \\ \beta & \text{(for row} \end{cases}$. In practice $\tilde{\beta} = T(JJ, J)$, $\tilde{\gamma} = T(J, JJ)$ and the indices J and JJ are given the values of $J1$ and $J2$ in the appropriate order.

Type 4. *single-single* (E *is* 1×1, D *is* 1×1). $v_1 = r_1(j)/\text{den}$, den $= \lambda - \alpha$.

Type 5. *formula breakdown.* If in any of the previous cases $D = E$ then the formulas for solution break down. There are two cases to consider.

(i) Linear Independence. Any element v_j for which the formula yields 0/0 can be set to any value, the most convenient is 0. This represents the existence of a whole space of eigenvectors associated with E.

(ii) Defective Case. Any element v_j for which the formula yields a value exceeding 1/TOL will cause the condition number to exceed 1/TOL. If this case is detected computation is interrupted, the condition number is set to 1/TOL, and the program proceeds to the next eigenvalue.

We propose that TOL $= 10^{-30}$ will be suitable for most applications and most computers.

These tests make the code simple and machine independent. However, it is possible to devise matrices for which the given value 1/TOL for the condition is very misleading. We know of no failsafe procedure which does not involve deciding the rank of $T - \xi$ for all ξ in a neighborhood of λ. This is a costly, difficult, and often unrewarding task.

3.3 *Closed Form Solution for Equations of Type* 1. The equations to be solved are of the form $-DV + V\hat{E} = R$, where

$$D = \begin{pmatrix} \alpha & \beta \\ \gamma & \alpha \end{pmatrix}, \quad \hat{E} = \begin{pmatrix} \lambda & \mu \\ -\mu & \lambda \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix}.$$

The standardization of the block triangular matrix T forces the diagonal elements of D to be equal. This yields simple formulas for the elements of V.

Rewrite the equation as

$$(v_{11}\ v_{12}\ v_{21}\ v_{22})\begin{pmatrix} B & -\gamma I_2 \\ -\beta I_2 & B \end{pmatrix} = (r_{11}\ r_{12}\ r_{21}\ r_{22}) = r^{\mathrm{T}},$$

where

$$B = \begin{pmatrix} \lambda - \alpha & \mu \\ -\mu & \lambda - \alpha \end{pmatrix}.$$

Observe that

$$\begin{pmatrix} B & -\gamma I_2 \\ -\beta I_2 & B \end{pmatrix}\begin{pmatrix} B^{\mathrm{T}} & \gamma I_2 \\ \beta I_2 & B^{\mathrm{T}} \end{pmatrix} = \tau I_4 + 2\mu J_4,$$

where $\tau = (\lambda - \alpha)^2 + \mu^2 - \beta\gamma$, with $\beta\gamma < 0$,

$$J_4 = \begin{pmatrix} 0 & 0 & 0 & \gamma \\ 0 & 0 & -\gamma & 0 \\ 0 & \beta & 0 & 0 \\ -\beta & 0 & 0 & 0 \end{pmatrix},$$

and I_k is the $k \times k$ identity matrix. Further, $(\tau I_4 + 2\mu J_4)(\tau I_4 - 2\mu J_4) = [\tau^2 - 4\mu^2(-\beta\gamma)]I_4$. Hence

$$(v_{11}\ v_{12}\ v_{21}\ v_{22})(\tau^2 + 4\mu^2\beta\gamma) = r^{\mathrm{T}}\begin{pmatrix} B^{\mathrm{T}} & \gamma I_2 \\ \beta I_2 & B^{\mathrm{T}} \end{pmatrix}(\tau I_4 - 2\mu J_4)$$

$$= r^{\mathrm{T}}\begin{pmatrix} e & -f & \gamma g & -\gamma h \\ f & e & \gamma h & \gamma g \\ \beta g & -\beta h & e & -f \\ \beta h & \beta g & f & e \end{pmatrix}$$

where $d = \lambda - \alpha$, $e = d\tau$, $f = \mu(\tau + 2\beta\gamma)$, $g = \tau - 2\mu^2$, and $h = 2\,d\mu$. Note that

$$\tau^2 + 4\mu^2\beta\gamma = (d^2 + \mu^2 - \beta\gamma)^2 + 4\mu^2\beta\gamma$$

$$= g^2 + h^2$$

$$(= 0 \quad \text{if and only if} \quad \alpha = \lambda,\ \mu^2 = -\beta\gamma).$$

These same formulas will be valid for the row eigenvectors provided we exchange (β, γ) and $(\mu, -\mu)$.

The alternative to using this closed form solution is to code a special version of Gaussian elimination with pivoting. It is the pivoting which would lengthen the code considerably.

3.4 *The Condition Number of Conjugate Pairs of Eigenvalues.* Let $\lambda \pm i\mu$ be a complex pair of eigenvalues of the real Schur matrix T obtained by the QR algorithm. In the course of the algorithm the following real equations are solved for real n-vectors u_1, u_2, w_1, w_2:

$$T(u_1, u_2) = (u_1, u_2)\begin{pmatrix} \rho & \mu \\ -\mu & \rho \end{pmatrix}, \quad (w_1, w_2)^*T = \begin{pmatrix} \rho & \mu \\ -\mu & \rho \end{pmatrix}(w_1, w_2)^*. \tag{5}$$

Thus span (u_1, u_2) and span (w_1^*, w_2^*) are real invariant subspaces under T. However, $\{u_1, u_2\}$ and $\{w_1^*, w_2^*\}$ are very special bases of these spaces.

LEMMA. *With the notation given above, $u_1 \pm iu_2$ and $w_1^* \mp iw_2^*$ are the column and row eigenvectors belonging to $\lambda \pm i\mu$.*

PROOF. From (5),

$$Tu_1 = u_1\lambda - u_2\mu, \quad w_1^*T = \lambda w_1^* + \mu w_2^*,$$

$$Tu_2 = u_1\mu + u_2\lambda, \quad w_2^*T = -\mu w_1^* + \lambda w_2^*.$$

Hence

$$T(u_1 + iu_2) = u_1(\lambda + i\mu) + iu_2(i\mu + \lambda),$$

$$(w_1^* - iw_2^*)T = (\lambda + i\mu)w_1^* - (\lambda + i\mu)iw_2^*.$$

The eigenvectors for $\lambda - i\mu$ are obtained in the same way. □

Consequently,

$$\text{cond}\,(\lambda \pm i\mu) = \| u_1 + iu_2 \| \cdot \| w_1^* - iw_2^* \| / [w_1^* u_1 + w_2^* u_2 + i(w_1^* u_2 - w_2^* u_2)].$$

Use was made in the lemma of the quasi-triangular nature of T. A consequence of this form is that u_1 and w_1 can be packed into the same real n-vector with two overlapping elements as indicated:

$$\left.\begin{aligned} u_1^* &= (x, \ldots, x, p_i, q_i, 0, \ldots, 0) \\ w_i^* &= (0, \ldots, 0, \bar{p}_i, \bar{q}_i, x, \ldots, x) \end{aligned}\right\} \quad i = 1, 2.$$

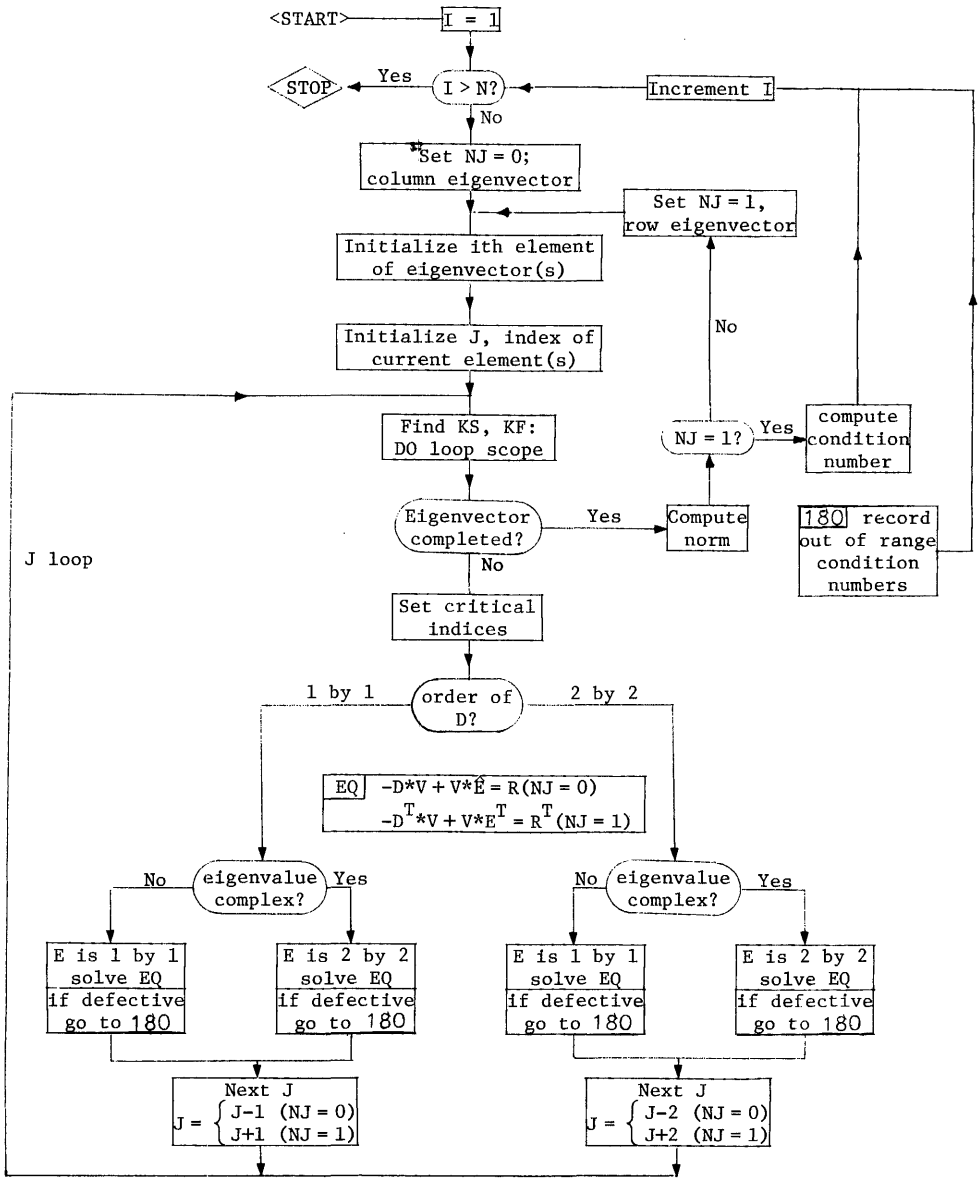


Fig. 1. Flowchart representing way in which column and row eigenvectors are computed allowing for haphazard ordering of 1×1 and 2×2 blocks on diagonal of real Schur form

The equations to be satisfied by p_i, q_i are of the form

$$\begin{pmatrix}\lambda & \beta\\ \gamma & \lambda\end{pmatrix}\begin{pmatrix}p_1 & p_2\\ q_1 & q_2\end{pmatrix} - \begin{pmatrix}p_1 & p_2\\ q_1 & q_2\end{pmatrix}\begin{pmatrix}\lambda & \mu\\ -\mu & \lambda\end{pmatrix} = 0,$$

$$\begin{pmatrix}\bar{p}_1 & \bar{q}_1\\ \bar{p}_2 & \bar{q}_2\end{pmatrix}\begin{pmatrix}\lambda & \beta\\ \gamma & \lambda\end{pmatrix} - \begin{pmatrix}\lambda & \mu\\ -\mu & \lambda\end{pmatrix}\begin{pmatrix}\bar{p}_1 & \bar{q}_1\\ \bar{p}_2 & \bar{q}_2\end{pmatrix} = 0,$$

where $\mu^2 = -\beta\gamma$. These equations reduce to

$$\beta q_2 = p_1\mu, \qquad \bar{p}_1\beta = \mu\bar{q}_2,$$

$$\beta q_1 = -\mu p_2, \qquad \beta\bar{p}_2 = -\mu\bar{q}_1.$$

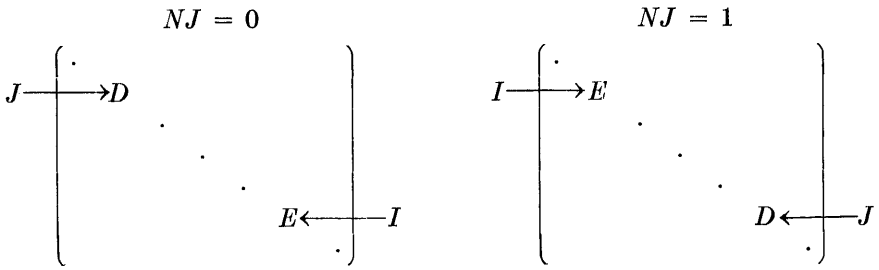
The simplest solution (which we adopt) takes $p_1 = \bar{p}_1 = 1$, $q_1 = p_2 = \bar{q}_1 = \bar{p}_2 = 0$, $q_2 = \mu/\beta$, $\bar{q}_2 = 1/q_2$. With this choice,

$$(w_1 - iw_2)^*(u_1 + iu_2) = (\bar{p}_1p_1 + \bar{p}_2p_2) + (\bar{q}_1q_1 + \bar{q}_2q_2) = 2 \text{ and}$$

$$\text{cond } (\lambda \pm i\mu) = [(\| u_1 \|^2 + \| u_2 \|^2)(\| w_1 \|^2 + \| w_2 \|^2)]^{1/2}/2. \tag{14}$$

4. The Structure of CONDIT

As indicated in Section 3.2 the same section of code (the J loop) computes the column and the row eigenvector of the Ith eigenvalue. It is the standardization of the Schur form which permits this economy. J, which always points to the block D, decreases for the column eigenvector ($NJ = 0$) and increases for the row eigenvector ($NJ = 1$), as indicated:



$$E = \begin{pmatrix}\lambda & \phi\\ \theta & \lambda\end{pmatrix} \text{ or } \lambda, \qquad \hat{E} = \begin{pmatrix}\lambda & \mu\\ -\mu & \lambda\end{pmatrix} \text{ or } \lambda,$$

where $\theta\phi = -\mu^2$.

Is the 0 diagonal element in $\mathbf{I}^0$ a keypunch error or did it really belong in the user's problem?

$$\mathbf{L}^0 = \begin{pmatrix}0 & \mathbf{X}\\ \mathbf{I}^0 & 0\end{pmatrix}; \quad \mathbf{I}^0 = \text{diag}(0, 1, 1, \ldots, 1);$$

$$\mathbf{X} = \begin{pmatrix}\mathbf{D} & \mathbf{T}_1 & 0\\ \mathbf{T}_2 & \mathbf{F}_1 & -\mathbf{F}_2\\ 0 & -\mathbf{F}_3 & \mathbf{F}_4\end{pmatrix}10^8; \quad \mathbf{F}\begin{pmatrix}a & b\\ c & d\end{pmatrix} = \begin{pmatrix}a & 0 & 0 & b\\ 0 & a & -b & 0\\ 0 & -c & d & 0\\ c & 0 & 0 & d\end{pmatrix};$$

$$\mathbf{D} = \text{diag}(.5221, .3563, .5552 \times 10^{-3}, .1328);$$

$$\mathbf{T}_1 = \begin{pmatrix}.5221 & 0 & 0 & -.8951\\ 0 & -.3563 & .6109 & 0\\ 0 & 0 & -.5552 \times 10^{-3} & 0\\ 0 & 0 & 0 & -.1328\end{pmatrix};$$

$$\mathbf{T}_2 = \begin{pmatrix}-.0976 & 0 & 0 & 0\\ 0 & -.0666 & 0 & 0\\ 0 & .02659 & -.4218 \times 10^{-4} & 0\\ -.03896 & 0 & 0 & -.1009 \times 10^{-1}\end{pmatrix}$$

$$\mathbf{F}_1 = \begin{pmatrix}2.859 & 0 & 0 & 1.079\\ 0 & 2.828 & -1.026 & 0\\ 0 & -.2389 & .4294 & 0\\ .2513 & 0 & 0 & .4607\end{pmatrix};$$

$$\mathbf{F}_2 = \mathbf{F}\begin{pmatrix}2.761 & .5891\\ .2123 & .3590\end{pmatrix}; \quad \mathbf{F}_3 = \mathbf{F}\begin{pmatrix}1.627 & .5373\\ .1096 & .2868\end{pmatrix}; \quad \mathbf{F}_4 = \mathbf{F}\begin{pmatrix}1.849 & .3731\\ .1178 & .4970\end{pmatrix};$$

$\| \mathbf{L}^0 \|_\infty \approx 5 \times 10^8$.

Fig. 2. 24X24 matrix for case study

Table I. Eigenvalues and Condition Numbers of $\mathbf{L}^0$, $\mathbf{L}$, $\mathbf{M}$

$\lambda_i(\mathbf{L}^0)$	Cond($\mathbf{L}$)	Cond ($\mathbf{L}^0$) (balanced)	$\lambda_i(\mathbf{L})$	Cond($\mathbf{L}$)	$\lambda_i(\mathbf{M})$	Cond($\mathbf{M}$)
$\pm.21534594 \times 10^5$	10^4	1	$\pm.21534594 \times 10^5$	10^4	$\pm.21534594 \times 10^5$	10^4
$\pm.18667890 \times 10^5$	10^4	10^3	$\pm.18654343 \times 10^5$	10^4	$\pm.18692513 \times 10^5$	10^4
$\pm.11076317 \times 10^5 i$	6×10^3	4×10^3	$\pm.1098663V \times 10^5 i$	6×10^3	$\pm.11142610 \times 10^6 i$	6×10^3
$\pm.82614552 \times 10^4$	6×10^3	5×10^3	$\pm.8646568V \times 10^4$	10^4	$\pm.86339960 \times 10^4$*	10^4
$\pm.83281998 \times 10^4$	6×10^3	1	$\pm.83281998 \times 10^4$	6×10^3	$\pm.83281998 \times 10^4$	6×10^3
$\pm.41438248 \times 10^4$	7×10^3	10^4	$\pm.7026706V \times 10^4$	10^4	$\pm.71883679 \times 10^4$*	10^4
$\pm.64209960 \times 10^4$	7×10^3	3	$\pm.64209960 \times 10^4$	6×10^3	$\pm.64209960 \times 10^4$	7×10^3
$\pm.33632953 \times 10^4$	5×10^3	9×10^3	$\pm.38448590 \times 10^4$	3×10^3	$\pm.38458962 \times 10^4$*	3×10^3
$\pm.35102700 \times 10^4$	2×10^3	4	$\pm.35102700 \times 10^4$	2×10^3	$\pm.35102700 \times 10^4$	2×10^3
$\pm.30530592 \times 10^4$	3×10^3	3	$\pm.30530592 \times 10^4$	3×10^3	$\pm.30530592 \times 10^4$	3×10^3
$\pm.0$†	10^{15}	10^{30}	$\pm.29241979 \times 10^4$	3×10^3	$\pm.29134631 \times 10^4$	3×10^3
$\pm.23559292 \times 10^3$	10^2	1	$\pm.23559291 \times 10^3$	10^2	$\pm.23559292 \times 10^3$	10^2

* The imaginary pair of eigenvalues had real parts less than 10^{-6} (a relative error of 10^{-11}).
V denotes a digit that changed when the matrix was balanced.
† The unbalanced matrix $\mathbf{L}^0$ had a negative eigenvalue -2×10^{-8} instead of -0.

At each major step in the J loop the equations EQ, given in the chart, are solved as D varies. Four cases are shown in the flowchart which is Figure 1.

5. Results

The matrix $\mathbf{L}^0$ described in Figure 2 came (in punched card form) from a large industrial company. It was causing their eigenvalue program to fail.

An inspection of the form of $\mathbf{L}^0$ suggests that the strange diagonal element in $\mathbf{I}^0$ and the discordant sign of the (1, 1) element of $\mathbf{T}_1$ were keypunch errors. Let us consider the matrices resulting from the removal of these anomalies:

$$\mathbf{L} = \begin{pmatrix} \mathbf{0} & \mathbf{X} \\ \mathbf{I} & \mathbf{0} \end{pmatrix}, \qquad \mathbf{M} = \begin{pmatrix} \mathbf{0} & \mathbf{Y} \\ \mathbf{I} & \mathbf{0} \end{pmatrix},$$

$$\mathbf{Y} = 10^8 \begin{pmatrix} \mathbf{D} & \mathbf{T}_1^- & \mathbf{0} \\ \mathbf{T}_2 & \mathbf{T}_1 & -\mathbf{F}_2 \\ \mathbf{0} & -\mathbf{F}_3 & \mathbf{F}_4 \end{pmatrix},$$

where $\mathbf{T}_1^-$ is obtained from $\mathbf{T}_1$ by reversing the sign of its (1, 1) element.

Notice that the eigenvalues of $\mathbf{L}$ are the square roots of those of $\mathbf{X}$:

$$\begin{pmatrix} \mathbf{0} & \mathbf{X} \\ \mathbf{I} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} \leftrightarrow \mathbf{X}\mathbf{v} = \lambda^2\mathbf{v}, \quad \mathbf{u} = \lambda\mathbf{v}.$$

The eigenvalues of $\mathbf{L}^0$, $\mathbf{L}$, and $\mathbf{M}$ are given in Table I and we offer the following comments. Every eigenvalue of $\mathbf{L}^0$ is moderately ill conditioned and the zero pair appear to belong to a quadratic elementary divisor (only one eigenvector). Perhaps some of this is due to the unbalanced nature of $\mathbf{L}^0$. The thirteenth row of $\mathbf{L}^0$ is null and this must be permuted out of the way before the rest of the matrix can

be balanced. The result is that none of the computed eigenvalues changed, but half of them became almost perfectly conditioned.

In fact we can say that the ill condition of all six pairs is due to the zero element in position (13, 1). When this is replaced by one, we obtain the matrix **L** which has six pairs of eigenvalues almost identical to the well conditioned pairs of the balanced $\mathbf{L}^0$. Four of the other six pairs are changed completely, the remaining two ($\pm.186 \times 10^5$ and $\pm.11 \times 10^5 i$) are substantially altered. Interestingly the balanced versions of **L** and **M** are almost normal and we have not bothered to record the condition numbers. The six pairs of eigenvalues which were unchanged by the move from $\mathbf{L}^0$ to **L** were also invariant in the change from **L** to **M**. The other six pairs had relative errors less than 2.5 percent.

We can tell in advance what the balanced form of **L** and **M** will be:

$$\hat{\mathbf{L}} = \begin{pmatrix} 0 & 10^{-4}X \\ 10^4 I & 0 \end{pmatrix}, \quad \hat{\mathbf{M}} = \begin{pmatrix} 0 & 10^{-4}Y \\ 10^4 I & 0 \end{pmatrix}.$$

The change from $\mathbf{L}^0$ to **L** is tiny relative to $\| \mathbf{L}^0 \|$ ($\doteq 10^{-8} \| \mathbf{L}^0 \|$) but the change from $\hat{\mathbf{L}}^0$ to $\hat{\mathbf{L}}$ is approximately $\| \hat{\mathbf{L}} \|$.

We conclude that the suspicious element in $\mathbf{L}^0$ was probably a keypunch error. Concerning the (1, 1) element of $\mathbf{T}_1$ we cannot say, both **L** and **M** are reasonable matrices and indeed the change of sign does not affect the leading two decimals in any eigenvalue.

REFERENCES

1. CHAN, S.P., FELDMAN, R., AND PARLETT, B. A program to compute the condition numbers of matrix eigenvalues without computing eigenvectors. Memo. No. ERL-M517, Electron. Res. Lab., College of Engineering, U. of California, Berkeley, Calif.
2. EISPACK Guide. Lecture Notes in Computer Science No. 6, Springer-Verlag, 1974.
3. International Mathematical and Statistical Library, Inc. (IMSL), 7500 Bellaire Blvd., Houston, Tex.
4. KAHAN, W. Conserving confluence curbs ill-condition. Tech. Rep. No. 6, Comptr. Sci. Dep., U. of California, Berkeley, Calif., 1972.
5. PARLETT, B.N., AND WANG, Y. The influence of the compiler on the cost of mathematical software—in particular on the cost of triangular factorization. *ACM Trans. on Math. Software 1*, 1 (March 1975), 35–46.
6. WILKINSON, J.H. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, 1975.
7. WILKINSON, J.H., AND REINSCH, C. *Handbook for Automatic Computation, II: Linear Algebra*. Springer-Verlag, 1971.

ALGORITHM

```
      SUBROUTINE QR2NOZ(NM, N, LOW, IGH, H, WR, WI, IERR)                QR2   1Ø
C PURPOSE                                                                QR2   2Ø
C THE FORTRAN SUBROUTINE QR2NOZ COMPUTES THE EIGENVALUES OF A REAL       QR2   3Ø
C      UPPER HESSENBERG MATRIX USING THE QR METHOD AND REDUCES THE       QR2   4Ø
C      MATRIX TO A STANDARDIZED QUASI-TRIANGULAR FORM.  COMPUTATIONS     QR2   5Ø
C      ARE DONE IN REAL ARITHMETIC.                                      QR2   6Ø
C THE SUBROUTINE STATEMENT IS                                            QR2   7Ø
C    SUBROUTINE QR2NOZ(NM,N,LOW,IGH,H,WR,WI,IERR).                       QR2   8Ø
C NM        IS AN INTEGER INPUT VARIABLE SET EQUAL TO THE ROW DIMENSION  QR2   9Ø
C           OF THE ARRAY H AS SPECIFIED IN THE CALLING PROGRAM.          QR2  1ØØ
C N         IS AN INTEGER INPUT VARIABLE SET EQUAL TO THE ORDER OF THE   QR2  11Ø
C           MATRIX H.  N  .LE.  NM                                       QR2  12Ø
C LOW,IGH   ARE INTEGER INPUT VARIABLES INDICATING THE BOUNDARY INDICES  QR2  13Ø
C           FOR THE BALANCED MATRIX.  IF THE MATRIX IS NOT BALANCED SET  QR2  14Ø
C           LOW TO 1 AND IGH TO N.                                       QR2  15Ø
C H         IS A REAL TWO-DIMENSIONAL ARRAY WITH ROW DIMENSION NM AND    QR2  16Ø
C           COLUMN DIMENSION AT LEAST N.  ON INPUT IT CONTAINS THE       QR2  17Ø
C           UPPER HESSENBERG MATRIX OF ORDER N.  ON OUTPUT IT CONTAINS   QR2  18Ø
C           THE STANDARDIZED QUASI-TRIANGULAR MATRIX.                    QR2  19Ø
C WR,WI     ARE REAL OUTPUT ONE-DIMENSIONAL VARIABLES OF DIMENSION AT     QR2  2ØØ
C           LEAST N CONTAINING THE REAL AND IMAGINARY PARTS,             QR2  21Ø
C           RESPECTIVELY, OF THE EIGENVALUES OF THE HESSENBERG MATRIX.   QR2  22Ø
C           THE EIGENVALUES ARE UNORDERED EXCEPT THAT COMPLEX CONJUGATE  QR2  23Ø
C           PAIRS OF EIGENVALUES APPEAR CONSECUTIVELY WITH THE           QR2  24Ø
C           EIGENVALUE HAVING THE POSITIVE IMAGINARY PART FIRST.         QR2  25Ø
```

```
C IERR      IS AN INTEGER OUTPUT VARIABLE SET EQUAL TO AN ERROR          QR2  26Ø
C           COMPLETION CODE.  IF MORE THAN 3Ø ITERATIONS ARE REQUIRED    QR2  27Ø
C           TO DETERMINE AN EIGENVALUE, THIS SUBROUTINE TERMINATES WITH  QR2  28Ø
C           IERR SET EQUAL TO THE INDEX OF THE EIGENVALUE FOR WHICH      QR2  29Ø
C           FAILURE OCCURS.  THE EIGENVALUES IN THE WR AND WI ARRAYS     QR2  3ØØ
C           SHOULD BE CORRECT FOR INDICES  IERR+1,IERR+2,...,N.  IF ALL  QR2  31Ø
C           THE EIGENVALUES ARE DETERMINED WITHIN 3Ø ITERATIONS, IERR    QR2  32Ø
C           IS SET TO ZERO.                                              QR2  33Ø
      DIMENSION H(NM,N), WR(N), WI(N)                                    QR2  34Ø
      REAL MACHEP                                                        QR2  35Ø
      INTEGER EN, ENM2                                                   QR2  36Ø
      LOGICAL NOTLAS                                                     QR2  37Ø
C MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING THE RELATIVE        QR2  38Ø
C PRECISION OF FLOATING POINT ARITHMETIC.  THE VALUE BELOW IS            QR2  39Ø
C EQUAL TO 2.**(-46), WHICH IS APPROPRIATE FOR CDC 6ØØØ-SERIES           QR2  4ØØ
C MACHINES.                                                              QR2  41Ø
      DATA MACHEP /16424ØØØØØØØØØØØØØØØØB/                              QR2  42Ø
      IERR = Ø                                                           QR2  43Ø
C STORE ROOTS ISOLATED BY BALANC                                         QR2  44Ø
      DO 1Ø I=1,N                                                        QR2  45Ø
        IF (I.GE.LOW .AND. I.LE.IGH) GO TO 1Ø                            QR2  46Ø
        WR(I) = H(I,I)                                                   QR2  47Ø
        WI(I) = Ø.Ø                                                      QR2  48Ø
   1Ø CONTINUE                                                           QR2  49Ø
      EN = IGH                                                           QR2  5ØØ
      T = Ø.Ø                                                            QR2  51Ø
C SEARCH FOR NEXT EIGENVALUES                                            QR2  52Ø
   2Ø IF (EN.LT.LOW) RETURN                                              QR2  53Ø
      ITS = Ø                                                            QR2  54Ø
      NA = EN - 1                                                        QR2  55Ø
      ENM2 = NA - 1                                                      QR2  56Ø
C LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT                             QR2  57Ø
C    FOR L=EN STEP -1 UNTIL LOW DO                                       QR2  58Ø
   3Ø IF (EN.EQ.LOW) GO TO 5Ø                                            QR2  59Ø
      DO 4Ø LL=LOW,NA                                                    QR2  6ØØ
        L = EN + LOW - LL                                                QR2  61Ø
        IF (ABS(H(L,L-1)).LE.MACHEP*(ABS(H(L-1,L-1))+ABS(H(L,L))))       QR2  62Ø
     *    GO TO 6Ø                                                      QR2  63Ø
   4Ø CONTINUE                                                           QR2  64Ø
   5Ø L = LOW                                                            QR2  65Ø
C FORM SHIFT                                                             QR2  66Ø
   6Ø X = H(EN,EN)                                                       QR2  67Ø
      IF (L.EQ.EN) GO TO 2ØØ                                             QR2  68Ø
      Y = H(NA,NA)                                                       QR2  69Ø
      W = H(EN,NA)*H(NA,EN)                                              QR2  7ØØ
      IF (L.EQ.NA) GO TO 21Ø                                             QR2  71Ø
      IF (ITS.EQ.3Ø) GO TO 27Ø                                           QR2  72Ø
      IF (ITS.NE.1Ø .AND. ITS.NE.2Ø) GO TO 8Ø                            QR2  73Ø
C FORM EXCEPTIONAL SHIFT                                                 QR2  74Ø
      T = T + X                                                          QR2  75Ø
      DO 7Ø I=LOW,EN                                                     QR2  76Ø
        H(I,I) = H(I,I) - X                                              QR2  77Ø
   7Ø CONTINUE                                                           QR2  78Ø
      S = ABS(H(EN,NA)) + ABS(H(NA,ENM2))                                QR2  79Ø
      X = Ø.75*S                                                         QR2  8ØØ
      Y = X                                                              QR2  81Ø
      W = -Ø.4275*S*S                                                    QR2  82Ø
   8Ø ITS = ITS + 1                                                      QR2  83Ø
C LOOK FOR TWO CONSECUTIVE SMALL SUB-DIAGONAL                            QR2  84Ø
C  ELEMENTS.  FOR M=EN-2 STEP -1 UNTIL L DO                             QR2  85Ø
      DO 9Ø MM=L,ENM2                                                    QR2  86Ø
        M = ENM2 + L - MM                                                QR2  87Ø
        ZZ = H(M,M)                                                      QR2  88Ø
        R = X - ZZ                                                       QR2  89Ø
        S = Y - ZZ                                                       QR2  9ØØ
        P = (R*S-W)/H(M+1,M) + H(M,M+1)                                  QR2  91Ø
        Q = H(M+1,M+1) - ZZ - R - S                                      QR2  92Ø
        R = H(M+2,M+1)                                                   QR2  93Ø
        S = ABS(P) + ABS(Q) + ABS(R)                                     QR2  94Ø
        P = P/S                                                          QR2  95Ø
        Q = Q/S                                                          QR2  96Ø
        R = R/S                                                          QR2  97Ø
        IF (M.EQ.L) GO TO 1ØØ                                            QR2  98Ø
        IF (ABS(H(M,M-1))*(ABS(Q)+ABS(R)).LE.MACHEP*ABS(P)*              QR2  99Ø
     *    (ABS(H(M-1,M-1))+ABS(ZZ)+ABS(H(M+1,M+1)))) GO TO 1ØØ           QR2 1ØØØ
   9Ø CONTINUE                                                           QR2 1Ø1Ø
```

```
  1ØØ MP2 = M + 2                                                    QR2 1Ø2Ø
      DO 11Ø I=MP2,EN                                                QR2 1Ø3Ø
        H(I,I-2) = Ø.Ø                                               QR2 1Ø4Ø
        IF (I.EQ.MP2) GO TO 11Ø                                      QR2 1Ø5Ø
        H(I,I-3) = Ø.Ø                                               QR2 1Ø6Ø
  11Ø CONTINUE                                                       QR2 1Ø7Ø
C DOUBLE QR STEP INVOLVING ROWS L TO EN                              QR2 1Ø8Ø
C   AND COLUMNS M TO EN.                                             QR2 1Ø9Ø
      DO 19Ø K=M,NA                                                  QR2 11ØØ
        NOTLAS = K.NE.NA                                             QR2 111Ø
        IF (K.EQ.M) GO TO 12Ø                                        QR2 112Ø
        P = H(K,K-1)                                                 QR2 113Ø
        Q = H(K+1,K-1)                                               QR2 114Ø
        R = Ø.Ø                                                      QR2 115Ø
        IF (NOTLAS) R = H(K+2,K-1)                                   QR2 116Ø
        X = ABS(P) + ABS(Q) + ABS(R)                                 QR2 117Ø
        IF (X.EQ.Ø.Ø) GO TO 19Ø                                      QR2 118Ø
        P = P/X                                                      QR2 119Ø
        Q = Q/X                                                      QR2 12ØØ
        R = R/X                                                      QR2 121Ø
  12Ø   S = SIGN(SQRT(P*P+Q*Q+R*R),P)                                QR2 122Ø
        IF (K.EQ.M) GO TO 13Ø                                        QR2 123Ø
        H(K,K-1) = -S*X                                              QR2 124Ø
        GO TO 14Ø                                                    QR2 125Ø
  13Ø   IF (L.NE.M) H(K,K-1) = -H(K,K-1)                             QR2 126Ø
  14Ø   P = P + S                                                    QR2 127Ø
        X = P/S                                                      QR2 128Ø
        Y = Q/S                                                      QR2 129Ø
        ZZ = R/S                                                     QR2 13ØØ
        Q = Q/P                                                      QR2 131Ø
        R = R/P                                                      QR2 132Ø
C ROW MODIFICATION                                                   QR2 133Ø
        DO 16Ø J=K,N                                                 QR2 134Ø
          P = H(K,J) + Q*H(K+1,J)                                    QR2 135Ø
          IF (.NOT.NOTLAS) GO TO 15Ø                                 QR2 136Ø
          P = P + R*H(K+2,J)                                         QR2 137Ø
          H(K+2,J) = H(K+2,J) - P*ZZ                                 QR2 138Ø
  15Ø     H(K+1,J) = H(K+1,J) - P*Y                                  QR2 139Ø
          H(K,J) = H(K,J) - P*X                                      QR2 14ØØ
  16Ø   CONTINUE                                                     QR2 141Ø
        J = MINØ(EN,K+3)                                             QR2 142Ø
C COLUMN MODIFICATION                                                QR2 143Ø
        DO 18Ø I=1,J                                                 QR2 144Ø
          P = X*H(I,K) + Y*H(I,K+1)                                  QR2 145Ø
          IF (.NOT.NOTLAS) GO TO 17Ø                                 QR2 146Ø
          P = P + ZZ*H(I,K+2)                                        QR2 147Ø
          H(I,K+2) = H(I,K+2) - P*R                                  QR2 148Ø
  17Ø     H(I,K+1) = H(I,K+1) - P*Q                                  QR2 149Ø
          H(I,K) = H(I,K) - P                                        QR2 15ØØ
  18Ø   CONTINUE                                                     QR2 151Ø
  19Ø CONTINUE                                                       QR2 152Ø
      GO TO 3Ø                                                       QR2 153Ø
C ONE ROOT FOUND                                                     QR2 154Ø
  2ØØ H(EN,EN) = X + T                                               QR2 155Ø
      WR(EN) = H(EN,EN)                                              QR2 156Ø
      WI(EN) = Ø.Ø                                                   QR2 157Ø
      EN = NA                                                        QR2 158Ø
      GO TO 2Ø                                                       QR2 159Ø
C TWO ROOTS FOUND                                                    QR2 16ØØ
  21Ø P = (Y-X)/2.Ø                                                  QR2 161Ø
      Q = P*P + W                                                    QR2 162Ø
      ZZ = SQRT(ABS(Q))                                              QR2 163Ø
      H(EN,EN) = X + T                                               QR2 164Ø
      X = H(EN,EN)                                                   QR2 165Ø
      H(NA,NA) = Y + T                                               QR2 166Ø
      IF (Q.LT.Ø.Ø) GO TO 22Ø                                        QR2 167Ø
      ZZ = P + SIGN(ZZ,P)                                            QR2 168Ø
C REAL PAIR                                                          QR2 169Ø
      WR(NA) = X + ZZ                                                QR2 17ØØ
      WR(EN) = WR(NA)                                                QR2 171Ø
      IF (ZZ.NE.Ø.Ø) WR(EN) = X - W/ZZ                               QR2 172Ø
      WI(NA) = Ø.Ø                                                   QR2 173Ø
      WI(EN) = Ø.Ø                                                   QR2 174Ø
      X = H(EN,NA)                                                   QR2 175Ø
      R = SQRT(X*X+ZZ*ZZ)                                            QR2 176Ø
      P = X/R                                                        QR2 177Ø
```

```
      Q = ZZ/R                                                          QR2 178Ø
      GO TO 23Ø                                                         QR2 179Ø
C COMPLEX PAIR                                                          QR2 18ØØ
  22Ø WR(NA) = X + P                                                    QR2 181Ø
      WR(EN) = X + P                                                    QR2 182Ø
      WI(NA) = ZZ                                                       QR2 183Ø
      WI(EN) = -ZZ                                                      QR2 184Ø
C MAKE DIAGONAL ELEMENTS EQUAL                                          QR2 185Ø
      IF (P.EQ.Ø.Ø) GO TO 26Ø                                           QR2 186Ø
      BPC = H(EN,NA) + H(NA,EN)                                         QR2 187Ø
      TX = SQRT(BPC*BPC+4.Ø*P*P)                                        QR2 188Ø
      Q = SQRT(.5*(1.Ø+ABS(BPC)/TX))                                    QR2 189Ø
      P = SIGN(P/(Q*TX),-BPC*P)                                         QR2 19ØØ
C ROW MODIFICATION                                                      QR2 191Ø
  23Ø DO 24Ø J=NA,N                                                     QR2 192Ø
        ZZ = H(NA,J)                                                    QR2 193Ø
        H(NA,J) = Q*ZZ + P*H(EN,J)                                      QR2 194Ø
        H(EN,J) = Q*H(EN,J) - P*ZZ                                      QR2 195Ø
  24Ø CONTINUE                                                          QR2 196Ø
C COLUMN MODIFICATION                                                   QR2 197Ø
      DO 25Ø I=1,N                                                      QR2 198Ø
        ZZ = H(I,NA)                                                    QR2 199Ø
        H(I,NA) = Q*ZZ + P*H(I,EN)                                      QR2 2ØØØ
        H(I,EN) = Q*H(I,EN) - P*ZZ                                      QR2 2Ø1Ø
  25Ø CONTINUE                                                          QR2 2Ø2Ø
  26Ø EN = ENM2                                                         QR2 2Ø3Ø
      GO TO 2Ø                                                          QR2 2Ø4Ø
  27Ø IERR = EN                                                         QR2 2Ø5Ø
      RETURN                                                            QR2 2Ø6Ø
      END                                                               QR2 2Ø7Ø

      SUBROUTINE CONDIT(NM, N, A, V1, V2, WI, COND)                     CON   1Ø
C  CONDIT COMPUTES THE CONDITION NUMBERS OF THE EIGENVALUES OF A        CON   2Ø
C       STANDARDIZED QUASI-TRIANGULAR MATRIX.                           CON   3Ø
C  THE SUBROUTINE STATEMENT IS                                          CON   4Ø
C                 SUBROUTINE CONDIT(NM,N,A,V1,V2,WI,COND).              CON   5Ø
C  ON INPUT                                                             CON   6Ø
C    NM      MUST BE SET TO THE ROW DIMENSION OF THE TWO DIMENSIONAL    CON   7Ø
C            ARRAY AS DECLARED IN THE CALLING PROGRAM.                  CON   8Ø
C    N       IS THE ORDER OF THE MATRIX.  N.LE.NM                       CON   9Ø
C    A       CONTAINS THE STANDARDIZED QUASI-TRIANGULAR MATRIX PRODUCED CON  1ØØ
C            BY QR2NOZ.                                                 CON  11Ø
C    WI      CONTAINS THE IMAGINARY PARTS OF THE EIGENVALUES.  THE      CON  12Ø
C            EIGENVALUES ARE UNORDERED EXCEPT THAT COMPLEX CONJUGATE    CON  13Ø
C            PAIRS APPEAR CONSECUTIVELY.                                CON  14Ø
C    V1,V2   ARE FOR TEMPORARY STORAGE.                                 CON  15Ø
C  ON OUTPUT                                                            CON  16Ø
C    A       IS UNALTERED.                                              CON  17Ø
C    COND    CONTAINS THE CONDITION NUMBERS CORRESPONDING TO THE        CON  18Ø
C            EIGENVALUES IN (V2,WI).  COND = 1./TOL IF THE USUAL        CON  19Ø
C            FORMULA WOULD CAUSE OVERFLOW OR YIELD A VALUE EXCEEDING    CON  2ØØ
C            1/TOL.  TOL NEED NOT DEPEND ON THE COMPUTER.               CON  21Ø
C    V2      CONTAINS THE REAL PARTS OF THE EIGENVALUES.                CON  22Ø
C                     TYPICAL USAGE                                     CON  23Ø
C      DIMENSION A(5Ø,5Ø),WR(5Ø),WI(5Ø),COND(5Ø),ORT(5Ø)                CON  24Ø
C ********************ENTER MATRIX A AND DIMENSIONS N,NM*****************CON  25Ø
C      LOW=1                                                            CON  26Ø
C      IGH=N                                                            CON  27Ø
C      CALL ORTHES(NM,N,LOW,IGH,A,ORT)                                  CON  28Ø
C      CALL QR2NOZ(NM,N,LOW,IGH,A,WR,WI,IERR)                           CON  29Ø
C      CALL CONDIT(NM,N,A,ORT,WR,WI,COND)                               CON  3ØØ
C ***************************************************************************CON  31Ø
C  NOTE THE USE OF ORT AND WR IN CONDIT                                 CON  32Ø
      DIMENSION A(NM,NM), V1(NM), V2(NM), WI(NM), COND(NM)              CON  33Ø
      DIMENSION R1(2), R2(2)                                            CON  34Ø
      DATA TOL /1.E-3Ø/                                                 CON  35Ø
C -------------------------------------------------------------------------CON  36Ø
      I = 1                                                             CON  37Ø
   1Ø IF (I.GT.N) GO TO 19Ø                                             CON  38Ø
      VALR = A(I,I)                                                     CON  39Ø
      VALI = WI(I)                                                      CON  4ØØ
      VALI2 = VALI*VALI                                                 CON  41Ø
C NJ GIVES EIGENVECTOR TYPE, Ø FOR COLUMN, 1 FOR ROW                    CON  42Ø
      NJ = Ø                                                            CON  43Ø
C INITIALIZE NONZERO ELEMENTS OF EIGENVECTOR (V1,V2)                    CON  44Ø
      V1(I) = 1.Ø                                                       CON  45Ø
```

```
      V2(I) = Ø.Ø                                                        CON  46Ø
   2Ø J = I - 1 + 2*NJ                                                   CON  47Ø
      IF (VALI.EQ.Ø.Ø) GO TO 3Ø                                          CON  48Ø
      V2(I+1) = VALI/A(I,I+1)                                            CON  49Ø
      V1(I+1) = Ø.Ø                                                      CON  5ØØ
      IF (NJ.EQ.1) V2(I+1) = 1.Ø/V2(I+1)                                 CON  51Ø
      J = I - 1 + 3*NJ                                                   CON  52Ø
C FIND THE INDICES OF ELEMENTS COMPUTED SO FAR                           CON  53Ø
   3Ø KS = J + 1 + NJ*(I-J-1)                                            CON  54Ø
      KF = I + 1 + NJ*(J-I-2)                                            CON  55Ø
      IF (VALI.EQ.Ø.Ø .AND. NJ.EQ.Ø) KF = KF - 1                         CON  56Ø
C TEST FOR COMPLETION OF EIGENVECTOR                                     CON  57Ø
      IF ((J+NJ.LT.1) .OR. (J+NJ.GT.N+1)) GO TO 13Ø                      CON  58Ø
C ***********************************************************************CON  59Ø
C *SOLVE -D*V + V*E = R FOR V = (V1,V2).  D IS A DIAGONAL BLOCK IN ROWS   CON  6ØØ
C * J1,J2, AND E IS THE REAL CANONICAL FORM OF THE ITH EIGENVALUE.       CON  61Ø
C * EITHER D OR E OR BOTH CAN BE  1 BY 1                                 CON  62Ø
C ***********************************************************************CON  63Ø
C  FIND J1 AND J2 (J1.LE.J2) FOR ALL CASES                               CON  64Ø
      JJ = J                                                             CON  65Ø
      IF (WI(J).NE.Ø.Ø) JJ = J - 1 + 2*NJ                                CON  66Ø
      JØ = NJ*(J-JJ)                                                     CON  67Ø
      J1 = JJ + JØ                                                       CON  68Ø
      J2 = J - JØ                                                        CON  69Ø
      D1 = VALR - A(J,J)                                                 CON  7ØØ
C CALCULATE RIGHT HAND SIDE R                                            CON  71Ø
      DO 7Ø L=J1,J2                                                      CON  72Ø
        LJ = L - J1 + 1                                                  CON  73Ø
        R1(LJ) = Ø.Ø                                                     CON  74Ø
        R2(LJ) = Ø.Ø                                                     CON  75Ø
        IF (VALI.NE.Ø.Ø) GO TO 5Ø                                        CON  76Ø
        DO 4Ø K=KS,KF                                                    CON  77Ø
          LK = NJ*(K-L)                                                  CON  78Ø
          AA = A(L+LK,K-LK)                                              CON  79Ø
          R1(LJ) = R1(LJ) + AA*V1(K)                                     CON  8ØØ
   4Ø   CONTINUE                                                         CON  81Ø
        GO TO 7Ø                                                         CON  82Ø
   5Ø   DO 6Ø K=KS,KF                                                    CON  83Ø
          LK = NJ*(K-L)                                                  CON  84Ø
          AA = A(L+LK,K-LK)                                              CON  85Ø
          R1(LJ) = R1(LJ) + AA*V1(K)                                     CON  86Ø
          R2(LJ) = R2(LJ) + AA*V2(K)                                     CON  87Ø
   6Ø   CONTINUE                                                         CON  88Ø
   7Ø CONTINUE                                                           CON  89Ø
      IF (JJ.NE.J) GO TO 1ØØ                                             CON  9ØØ
C ****************************D IS 1 BY 1 ******************************CON  91Ø
      IF (VALI.NE.Ø.Ø) GO TO 8Ø                                          CON  92Ø
C E IS 1 BY 1 (D IS 1 BY 1)                                              CON  93Ø
      IF (ABS(D1).LT.TOL*ABS(R1(1))) GO TO 18Ø                           CON  94Ø
      V1(J) = Ø.Ø                                                        CON  95Ø
      V2(J) = Ø.Ø                                                        CON  96Ø
      IF (D1.NE.Ø.Ø) V1(J) = R1(1)/D1                                    CON  97Ø
      GO TO 9Ø                                                           CON  98Ø
C E IS 2 BY 2 ( D IS 1 BY 1 )                                            CON  99Ø
   8Ø DEN = D1*D1 + VALI2                                                CON 1ØØØ
      VAL = VALI*(-1.Ø)**NJ                                              CON 1Ø1Ø
      V1(J) = R1(1)*D1 + R2(1)*VAL                                       CON 1Ø2Ø
      V2(J) = R2(1)*D1 - R1(1)*VAL                                       CON 1Ø3Ø
      VMAX = AMAX1(ABS(V1(J)),ABS(V2(J)))                                CON 1Ø4Ø
      IF (DEN.LT.TOL*VMAX) GO TO 18Ø                                     CON 1Ø5Ø
      V1(J) = V1(J)/DEN                                                  CON 1Ø6Ø
      V2(J) = V2(J)/DEN                                                  CON 1Ø7Ø
C NEXT J                                                                 CON 1Ø8Ø
   9Ø J = J - 1 + 2*NJ                                                   CON 1Ø9Ø
      GO TO 3Ø                                                           CON 11ØØ
C ******************************D IS 2 BY 2*****************************CON 111Ø
  1ØØ IF (VALI.NE.Ø.Ø) GO TO 11Ø                                         CON 112Ø
C E IS 1 BY 1 (D IS 2 BY 2)                                              CON 113Ø
      DEN = D1*D1 + WI(J)**2                                             CON 114Ø
      V2(J1) = Ø.Ø                                                       CON 115Ø
      V2(J2) = Ø.Ø                                                       CON 116Ø
      V1(J1) = R1(1)*D1 + R1(2)*A(JJ,J)                                  CON 117Ø
      V1(J2) = R1(1)*A(J,JJ) + R1(2)*D1                                  CON 118Ø
      VMAX = AMAX1(ABS(V1(J1)),ABS(V1(J2)))                              CON 119Ø
      IF (DEN.LT.TOL*VMAX) GO TO 18Ø                                     CON 12ØØ
      V1(J1) = V1(J1)/DEN                                                CON 121Ø
```

```
      V1(J2) = V1(J2)/DEN                                                    CON 122Ø
      GO TO 12Ø                                                              CON 123Ø
C E IS 2 BY 2 (D IS 2 BY 2).  CLOSED FORM SOLUTION                           CON 124Ø
 11Ø B = A(JJ,J)                                                             CON 125Ø
      C = A(J,JJ)                                                            CON 126Ø
      VAL = VALI*(-1.Ø)**NJ                                                  CON 127Ø
      BXC = B*C                                                              CON 128Ø
      H = D1*D1 + VALI2 - BXC                                                CON 129Ø
      E = D1*H                                                               CON 13ØØ
      F = VAL*(H+2.Ø*BXC)                                                    CON 131Ø
      G = H - 2.Ø*VALI2                                                      CON 132Ø
      H = 2.Ø*D1*VAL                                                         CON 133Ø
      V1(J1) = R1(1)*E + R2(1)*F + R1(2)*B*G + R2(2)*B*H                     CON 134Ø
      V2(J1) = -R1(1)*F + R2(1)*E - R1(2)*B*H + R2(2)*B*G                    CON 135Ø
      V1(J2) = R1(1)*C*G + R2(1)*C*H + R1(2)*E + R2(2)*F                     CON 136Ø
      V2(J2) = -R1(1)*C*H + R2(1)*C*G - R1(2)*F + R2(2)*E                    CON 137Ø
      VMAX = AMAX1(ABS(V1(J1)),ABS(V2(J1)),ABS(V1(J2)),ABS(V2(J2)))          CON 138Ø
      DEN = G*G + H*H                                                        CON 139Ø
      IF (DEN.LT.TOL*VMAX) GO TO 18Ø                                         CON 14ØØ
      IF (DEN.EQ.Ø.Ø) GO TO 12Ø                                              CON 141Ø
      V1(J1) = V1(J1)/DEN                                                    CON 142Ø
      V2(J1) = V2(J1)/DEN                                                    CON 143Ø
      V1(J2) = V1(J2)/DEN                                                    CON 144Ø
      V2(J2) = V2(J2)/DEN                                                    CON 145Ø
C NEXT J                                                                     CON 146Ø
 12Ø J = J - 2 + 4*NJ                                                        CON 147Ø
      GO TO 3Ø                                                               CON 148Ø
C COMPUTE EIGENVECTOR NORM                                                   CON 149Ø
 13Ø VMAX = Ø.Ø                                                              CON 15ØØ
      DO 14Ø K=KS,KF                                                         CON 151Ø
        VMAX = VMAX + V1(K)**2 + V2(K)**2                                    CON 152Ø
 14Ø CONTINUE                                                                CON 153Ø
      IF (NJ.EQ.1) GO TO 15Ø                                                 CON 154Ø
C PREPARE TO COMPUTE ROW EIGENVECTOR                                         CON 155Ø
      NJ = 1                                                                 CON 156Ø
      CNORM2 = VMAX                                                          CON 157Ø
      GO TO 2Ø                                                               CON 158Ø
C COMPUTE CONDITION NUMBER                                                   CON 159Ø
 15Ø COND(I) = SQRT(CNORM2*VMAX)                                             CON 16ØØ
 16Ø IF (VALI.EQ.Ø.Ø) GO TO 17Ø                                              CON 161Ø
      COND(I) = COND(I)/2.Ø                                                  CON 162Ø
      COND(I+1) = COND(I)                                                    CON 163Ø
      I = I + 1                                                              CON 164Ø
C NEXT I                                                                     CON 165Ø
 17Ø I = I + 1                                                               CON 166Ø
      GO TO 1Ø                                                               CON 167Ø
C DEFECTIVE CASE                                                             CON 168Ø
 18Ø COND(I) = 1.Ø/TOL                                                       CON 169Ø
      GO TO 16Ø                                                              CON 17ØØ
C PLACE REAL PART OF EIGENVALUE IN V2                                        CON 171Ø
 19Ø DO 2ØØ I=1,N                                                            CON 172Ø
        V2(I) = A(I,I)                                                       CON 173Ø
 2ØØ CONTINUE                                                                CON 174Ø
      RETURN                                                                 CON 175Ø
      END                                                                    CON 176Ø
```

ALGORITHM 518
Incomplete Bessel Function I_0: The Von Mises Distribution [S14]

GEOFFREY W. HILL

Centre de Morphologie Mathématique, Fontainebleau, France

Key Words and Phrases: incomplete Bessel function, von Mises distribution, direction angles, circular statistics, asymptotic normal approximation
CR Categories: 5.5, 5.12
Language: Fortran

DESCRIPTION

The incomplete modified Bessel function $I_0(\Theta, \kappa)$ is formally equivalent to the cumulative distribution applied by von Mises [9] to study deviations of atomic weights from integer values, representable as points on the circumference of a circle or as circular directions. This distribution of points on a circle is analogous to the normal or Gaussian distribution of points on a line and has applications to the study of quantal or periodic data, directions of sedimentary bedding, surface fault lines, wildlife movements, etc. (cf. Batschelet [3] and Mardia [7]). The left tail area of this symmetrical distribution is evaluated by this Fortran FUNCTION of the angular deviation Θ and the concentration parameter κ, where

$$I_0(\Theta, \kappa) = P(x \leq \Theta; \kappa) = [2\pi I_0(\kappa)]^{-1} \int_{-\pi}^{\theta} \exp(\kappa \cos x)\, dx, \quad -\pi \leq \Theta < \pi.$$

A compromise between generality and efficiency is achieved by parameterizing the algorithm so that versions corresponding to different levels of accuracy are provided by selecting parameter sets as displayed in Table I.

The method of calculation for small κ involves backwards recursion through a series expansion in terms of modified Bessel functions, while for large κ an auxiliary FUNCTION for the normal probability integral or for the error function erf (x) is applied to an asymptotic normal approximation to order κ^{-4}.

In the case of small κ the series expansion given by Gumbel et al. [6] can be expressed as

$$\begin{aligned} F(\Theta; \kappa) &= \tfrac{1}{2} + \Theta/(2\pi) + \{\pi I_0(\kappa)\}^{-1} \sum_{n=1}^{\infty} n^{-1} I_n(\kappa) \sin(n\Theta) \\ &= \tfrac{1}{2} + \Theta/(2\pi) + \pi^{-1} V_1, \end{aligned}$$

where $V_1 = R_1(\kappa)\,[\sin\Theta + R_2(\kappa)\,[2^{-1}\sin 2\Theta + R_3(\kappa)\,[3^{-1}\sin 3\Theta + \cdots]]]$ represents a "nested" expression in terms of $R_n(\kappa) = I_n(\kappa)/I_{n-1}(\kappa)$, $n = 1, 2, 3, \ldots,$

Received 12 December 1975 and 8 June 1976.

This work was supported by a French Government Scientific Fellowship and a grant of computer time on the IRIS 80 of the École des Mines de Paris.
Author's present address: Division of Mineral Chemistry, CSIRO, Port Melbourne, Australia.

Table I. Parameter Values for Several Accuracy Levels

Accuracy in decimal digits	Recurrence over $[a_1 + a_2\kappa - a_3/(\kappa + a_4)]$ terms of nested series				Critical value of κ	Normalizing approximation
(D)	a_1	a_2	a_3	a_4	(CK)	(C_1)
6	8	1.0	3.0	1.0	6.5	62.0
7	9	1.0	3.5	0.7	8.0	60.5
8	12	0.8	8	1.0	10.5	56.0
9	15	0.75	16	1.5	15	53.0
10	18	0.7	24	2.0	21	51.7
11	22	0.6	48	3.5	32	50.8
12	28	0.5	100	5.0	50	50.1

the modified Bessel function ratio denoted as r_{n-1} by Amos [2]. Recursive evaluation of the Bessel function ratio is known to be numerically unstable in the direction of increasing n, but backwards recursion generates additional terms of the series expansions in powers of κ, so that numerical stability is achieved for $n = p - 1, \ldots, 3, 2, 1$, from the formulation:

$$\sin n\Theta = \sin (n + 1)\Theta \cos \Theta - \cos (n + 1)\Theta \sin \Theta$$

$$\cos n\Theta = \cos (n + 1)\Theta \cos \Theta + \sin (n + 1)\Theta \sin \Theta$$

$$R_n(\kappa) = [2n/\kappa + R_{n+1}(\kappa)]^{-1} \qquad \text{(cf. [2, eq. (2)])}$$

$$V_n = R_n(\kappa)\,[\sin (n\Theta)/n + V_{n+1}].$$

Starting values can be computed directly for $\sin (p\Theta)$ and $\cos (p\Theta)$, while $R_p(\kappa)$ and V_p may be assigned zero values, provided p is large enough, since $I_p(\kappa)/I_0(\kappa) \leq (\kappa/2)^p/p!$ implies that the truncation error of V_1 will be less than the corresponding truncation error of the convergent series expansion of $\exp (\kappa/2)$. An upper bound on the truncation error was calculated for various combinations of values of κ and p. Curves of $p(\kappa; D)$, interpolated for assigned accuracy levels of D decimal places, were found to be conservatively bounded by relations of the form $p = [a_1 + a_2\kappa - a_3/(\kappa + a_4)]$, where suitable values of coefficients $a_1, a_2, \ldots,$ are shown in Table I. Since larger values of κ entail larger values of p and thus more computing time and accumulated roundoff error, an alternative method is desired when κ is larger than some critical value, which we denote by CK.

For large values of κ the asymptotic normality of the von Mises distribution may be exploited in the following three steps. First, the angle Θ is transformed to the nearly normally distributed variate $z = b(\kappa) \sin (\Theta/2)$, where $b(\kappa) = (2/\pi)^{1/2} \exp (\kappa)/I_0(\kappa)$ is conveniently computed from a continued fraction approximation:

$$b^2(\kappa) \doteq (C - 6 - 54/(C - 26 - 347/(C - C_1)))/6, \quad C = 24\kappa.$$

Values for C_1, fitted for the minimum value of $\kappa = CK$ are included in Table I. The second step uses an approximation to order κ^{-4} for the asymptotic normalizing series in terms of z [4]:

$$\chi \doteq z - z^3\left\{(C - 2z^2 - 16)/3 - \left(z^4 + \frac{7}{4}z^2 + \frac{167}{2}\right) \Big/ (C - C_1 - z^2 + 3)\right\}^{-2}$$

in which the same value of C_1 is used in the divisor to reduce the remaining error of order κ^{-5}. The third step calls an auxiliary function, GAUSS (χ), to evaluate the normal integral

$$\Phi(\chi) = (2\pi)^{-1/2} \int_{-\infty}^{\chi} \exp (-t^2/2)\, dt$$

which, by virtue of the normalizing transforms, provides the required value of the von Mises integral, $F(\Theta; \kappa) = \Phi(\chi)$. Alternatively the normal integral may be

evaluated in terms of the error function, erf (x), by virtue of the relationship, $\Phi(x) = \frac{1}{2}[1 + \text{erf}\ (x/\sqrt{2})]$, where erf $(-x) = -\text{erf}\ (x)$.

The error of the asymptotic approximation decreases rapidly with increasing κ and can be calculated by comparison with sufficiently accurate results from the backwards recursion method. In this way critical values of CK in Table I were obtained such that for $\kappa > CK$ the asymptotic approximation is accurate to at least D decimal digits. Since the number of arithmetic operations for the recursive evaluation increases with increasing κ beyond that required for the asymptotic approximation and the auxiliary function, it is efficient to use backwards recursion for $\kappa \leq CK$ and asymptotic approximation for $\kappa > CK$.

Tests of the algorithm using values from Table I show that resultant probability values are accurate to at least D decimal digits for backward recurrence over $[a_1 + a_2\kappa - a_3/(\kappa + a_4)]$ terms when $\kappa \leq CK$ and for the normal approximations using the corresponding value of C_1 when $\kappa > CK$. For other accuracy levels interpolation should provide useful estimates which may need adjustment to improve efficiency without loss of desired accuracy. The level of accuracy achievable is limited by that of the standard functions for SIN, COS, and SQRT or of the auxiliary function, GAUSS or ERF. For lower-accuracy versions little is lost by use of a linear bound, $p = \kappa + \frac{4}{3}(D - \frac{1}{2})$, but for higher-accuracy versions the tighter bounding curve is more efficient. Should very high accuracy be required, it would be more efficient to utilize high order terms in the asymptotic series [4] to enable reduction of CK.

For real values of Θ outside the range $(-\pi, +\pi)$ the result from recursive evaluation lies outside the range (0,1), whereas the normalizing approximation yields results in the required range but lacks the monotonic increasing property appropriate for a cumulative distribution function. It is convenient to extend the domain of definition over all real Θ by treating the angle as reduced modulo 2π to the range $(-\pi, +\pi)$, so that results for both small κ and large κ remain consistent with the properties of a probability distribution. With this convention for angles in a circle, the probability over an interval (Θ_1, Θ_2), evaluated as $F(\Theta_2; \kappa) - F(\Theta_1; \kappa)$, can be negative if the interval includes $(2n + 1)\pi$, in which case the correct result is obtained by adding 1. Because accumulated roundoff error can produce results lying just outside (0,1) in extreme cases, such results are replaced by the limit values 0 or 1. It is also convenient to treat negative values of κ as zero so that the result (the cumulative uniform distribution if $\kappa \leq 0$) remains defined for all real κ as well as for all real Θ, and thus no error conditions can arise from invalid actual parameter values.

The algorithm is presented in the form of a real-valued FUNCTION VMISES (T, VK). T is the real value of the angle Θ in radians, VK is the real value of the concentration parameter κ. This form requires an auxiliary function GAUSS(X) having a single real argument and returning the left tail area of the normal distribution as a real value in the interval (0,1). A suitable function accurate to 10 decimal places can be provided by Algorithm CJ39 [1]. For higher-accuracy levels the somewhat slower algorithm, ACM Algorithm 304 [5] would return the required value with a precision matched to the processor used.

If an auxiliary real function, ERF(X), is supplied instead of GAUSS(X), columns 7 to 50 of each of the last three comment cards marked IF ERF must replace the statement preceding it, thus providing an appropriate alternative version of the function VMISES.

The backwards recurrence method in this algorithm is considerably faster and requires less storage than Algorithm AS 86 [8] which evaluates the von Mises distribution with accuracy up to 8D for Θ values extended to the whole real line. The approximation, accurate to 3D, offered for $\kappa > 30$ in the description of Algorithm AS 86, is considerably less effective than the asymptotic approximation in this algorithm.

The performance of this algorithm was tested on an IRIS 80 using a Fortran compiler with single precision to about 7 decimals and double precision to about

16 decimals. The numerical stability of backwards recursive evaluation of sin $n\Theta$ and cos $n\Theta$ was tested in single and double precision for $p \leq 50$, $n = 1$, indicating accuracy loss of at most 2 decimal digits. To check the backwards recurrence method, values of $1 - 2P(\Theta; \kappa)$ were computed by the algorithm for $\kappa = 0(0.2)4$ and $\Theta = 0°(5°)\ 180°$ and compared with Table 4 of Gumbel et al. [6]. Results corresponded within a unit in the last place, except for two misprints in the published tables; $\Phi_\alpha(\alpha = 10°, \kappa = 1.8) = 0.15740$ should be 0.16740 and $\Phi_\alpha(\alpha = 40°, \kappa = 4.0) = 0.71123$ should be 0.81123. The validity of the asymptotic approximation was tested by comparison with results of extensive recursion for a range of values of Θ and κ. Tests with negative and zero values of κ and Θ near 0 and 2π and in the range $(-5\pi, +5\pi)$ confirmed performance of the algorithm as specified.

REFERENCES

1. Adams, A.G. Algorithm 39. Areas under the normal curve. *Comptr. J. 12* (1969), 197–198.
2. Amos, D.E. Computation of modified Bessel functions and their ratios. *Math. Comput. 28* (1974), 239–251.
3. Batschelet, E. Statistical methods for the analysis of problems in animal orientation and certain biological rhythms. Monograph, Amer. Inst. Biol. Sci., Washington, D.C., 1965.
4. Hill, G.W. New approximations to the von Mises distribution. *Biometrika 63*, 3 (1976), 673–676.
5. Hill, I.D., and Joyce, S.A. Algorithm 304. Normal curve integral. *Comm. ACM 10*, 6 6 (June 1967), 374–375.
6. Gumbel, E.J., Greenwood, J.A., and Durand, D. The circular normal distribution: theory and tables. *J. Amer. Statist. Assn. 48* (1953), 131–152.
7. Mardia, K.V. *Statistics of Directional Data.* Academic Press, New York, 1972.
8. Mardia, K.V., and Zemrock, P.J. Algorithm AS 86. The von Mises distribution function. *Appl. Statist. 24* (1975), 268–272.
9. Von Mises, R. Uber die "Ganzzahlighkeit" der Atomgewicht und verwandte Fragen. *Physikalische Z. 19* (1918), 490–500.

ALGORITHM

```
      FUNCTION VMISES(T, VK)                                          VMI   1Ø
C VMISES RETURNS THE LEFT TAIL AREA OF THE VON MISES DISTRIBUTION,     VMI   2Ø
C EQUAL TO THE INCOMPLETE MODIFIED BESSEL FUNCTION,  OF THE FIRST     VMI   3Ø
C KIND AND ZERO-TH ORDER.                                             VMI   4Ø
C     T  = ANGLE IN RADIANS, TREATED AS DEVIATION FROM ZERO (MEAN)    VMI   5Ø
C            REDUCED MODULO 2PI TO THE RANGE (-PI,+PI).               VMI   6Ø
C     VK = CONCENTRATION PARAMETER, KAPPA.    NEGATIVE VALUES ARE     VMI   7Ø
C            TREATED AS ZERO.                                         VMI   8Ø
C NEEDS AS AUXILIARY ROUTINE EITHER                                   VMI   9Ø
C FUNCTION GAUSS(X),  RETURNING THE NORMAL DISTRIBUTION TAIL AREA     VMI  1ØØ
C                        TO THE LEFT OF THE BOUNDING ORDINATE, X,     VMI  11Ø
C OR  FUNCTION ERF(X),  RETURNING THE ERROR FUNCTION AREA BETWEEN     VMI  12Ø
C      ZERO AND THE BOUNDING ORDINATE, X, WHERE ERF(-X) = -ERF(X).    VMI  13Ø
C      IF ERF IS USED THEN EACH OF THE LAST 3 COMMENTS (COLUMNS 7     VMI  14Ø
C      TO 5Ø ONLY) MUST REPLACE THE STATEMEMT PRECEDING IT.           VMI  15Ø
      DATA PI /3.1415926535898/, TPI /6.2831853Ø7176Ø/                VMI  16Ø
C CONSTANTS APPROPRIATE FOR 8 DECIMAL DIGIT ACCURACY.                 VMI  17Ø
      DATA A1, A2, A3, A4, CK, C1 /12.Ø,Ø.8,8.Ø,1.Ø,1Ø.5,56.Ø/        VMI  18Ø
      Z = VK                                                          VMI  19Ø
C CONVERT ANGLE T MODULO 2PI TO RANGE (-PI,+PI).                      VMI  2ØØ
      U = AMOD(T+PI,TPI)                                              VMI  21Ø
      IF (U.LT.Ø.Ø) U = U + TPI                                       VMI  22Ø
      Y = U - PI                                                      VMI  23Ø
      IF (Z.GT.CK) GO TO 3Ø                                           VMI  24Ø
      V = Ø.Ø                                                         VMI  25Ø
      IF (Z.LE.Ø.Ø) GO TO 2Ø                                          VMI  26Ø
C FOR SMALL VK SUM IP TERMS BY BACKWARDS RECURSION.                   VMI  27Ø
      IP = Z*A2 - A3/(Z+A4) + A1                                      VMI  28Ø
      P = FLOAT(IP)                                                   VMI  29Ø
      S = SIN(Y)                                                      VMI  3ØØ
      C = COS(Y)                                                      VMI  31Ø
      Y = P*Y                                                         VMI  32Ø
      SN = SIN(Y)                                                     VMI  33Ø
      CN = COS(Y)                                                     VMI  34Ø
      R = Ø.Ø                                                         VMI  35Ø
      Z = 2.Ø/Z                                                       VMI  36Ø
      DO 1Ø N=2,IP                                                    VMI  37Ø
```

```
      P = P - 1.Ø                                                    VMI  38Ø
      Y = SN                                                         VMI  39Ø
      SN = SN*C - CN*S                                               VMI  4ØØ
      CN = CN*C + Y*S                                                VMI  41Ø
      R = 1.Ø/(P*Z+R)                                                VMI  42Ø
      V = (SN/P+V)*R                                                 VMI  43Ø
   1Ø CONTINUE                                                       VMI  44Ø
   2Ø VMISES = (U*Ø.5+V)/PI                                          VMI  45Ø
      GO TO 4Ø                                                       VMI  46Ø
C FOR LARGE VK COMPUTE THE NORMAL APPROXIMATION AND LEFT TAIL.       VMI  47Ø
   3Ø C = 24.Ø*Z                                                     VMI  48Ø
      V = C - C1                                                     VMI  49Ø
      R = SQRT((54.Ø/(347.Ø/V+26.Ø-C)-6.Ø+C)/6.Ø)                    VMI  5ØØ
C     R=SQRT((54.Ø/(347.Ø/V+26.Ø-C)-6.Ø+C)/12.Ø)          IF ERF     VMI  51Ø
      Z = SIN(Y*Ø.5)*R                                               VMI  52Ø
      S = Z*Z                                                        VMI  53Ø
C     S=Z*Z*2.Ø                                           IF ERF     VMI  54Ø
      V = V - S + 3.Ø                                                VMI  55Ø
      Y = (C-S-S-16.Ø)/3.Ø                                           VMI  56Ø
      Y = ((S+1.75)*S+83.5)/V - Y                                    VMI  57Ø
      VMISES = GAUSS(Z-S/(Y*Y)*Z)                                    VMI  58Ø
C     VMISES=ERF(Z-S/(Y*Y)*Z)*Ø.5+Ø.5                     IF ERF     VMI  59Ø
   4Ø IF (VMISES.LT.Ø.Ø) VMISES = Ø.Ø                                VMI  6ØØ
      IF (VMISES.GT.1.Ø) VMISES = 1.Ø                                VMI  61Ø
      RETURN                                                         VMI  62Ø
      END                                                            VMI  63Ø
```

519-P 1- 0

ALGORITHM 519
Three Algorithms for Computing Kolmogorov-Smirnov Probabilities With Arbitrary Boundaries and a Certification of Algorithm 487 [S14]

RALPH KALLMAN
Ball State University

Key Words and Phrases: Kolmogorov-Smirnov probabilities, sample distribution function, boundary crossing probability
CR Categories: 5.5
Language: Fortran

DESCRIPTION

Introduction

Let $A(y)$, $B(y)$ be nondecreasing functions on [0, 1] where $B(0) < 0 < A(0)$, $B(1) < 1 < A(1)$. Let $G_n(y)$ be the sample distribution function for n independent random variables uniform on [0, 1]. We present three methods for computing $P = \Pr\{B(y) < G_n(y) < A(y), y \in [0, 1]\}$. The special case where $A(y) = y + D$, $B(y) = y - D$ has been treated by ACM Algorithm 487 [4].

Subroutine RAKK. A Generalization of Massey's Method

Massey has given a recursion formula [3; 5, p. 341, eq. (11.7.9)] for computing P when $A(y) = y + k/n$, $B(y) = y - k/n$. This can be generalized to arbitrary boundaries as follows. Let $y(1, j)$, $y(2, j)$ be the level points of $B(y)$, $A(y)$ for the ordinates $(j - 1)/n$ (e.g. $A(y(2, j)) = (j - 1)/n$). If the boundaries are not continuous and strictly increasing, let

$$y(2,j) = \sup\{y \in [0,1] \mid A(y) \leq (j-1)/n\}$$

$$y(1,j) = \inf(y \in [0,1] \mid B(y) \geq (j-1)/n\}$$

for those j such that the sup or inf is taken over a nonempty set. Arrange the $y(k, j)$ into a single ordered sequence $y(i)$, $i = 1, \ldots, m$, where $y(1) = 0$, $y(m) = 1$ are attached if not already present. Let $jt(i)$ = largest integer j such that $(j - 1)/n < A(y(i) -)$ and $jb(i)$ = smallest integer j such that $(j - 1)/n > B(y(i) +)$, $1 < i < m$. Note that if $jt(m) < n + 1$, then $P = 0$. Then define $j\text{top}(i) = \min(jt(i), n + 1)$, $j\text{bot}(i) = \max(jb(i), 1)$, $1 < i < m$; $j\text{bot}\ (1) = j\text{top}\ (1) = 1$; $j\text{bot}\ (m) = j\text{top}\ (m) = n + 1$. Then

$$P = \sum_j n! \prod_{k=2}^{m} \frac{[y(k) - y(k-1)]^{j(k)-j(k-1)}}{j(k)!}$$

Received 29 July 1975, 8 June 1976, and 14 September 1976.

Author's address: Department of Mathematical Sciences, Ball State University, Muncie, IN 47306.

where the summation is over the index set $\{j \equiv (j(1), \ldots, j(m)) \mid j\text{bot}(i) \leq j(i) \leq j\text{top}(i)\}$. Note that $j(1) \equiv 1$.

The computer program RAKK sums these multinomial probabilities recursively by computing $W(k, i)$ where

$$W(1, 1) = 1$$

$$W(k, i) = \sum W(j, i-1)\,[y(i) - y(i-1)]^{k-j}/(k-j)!,$$

$$k = j\text{bot}(i), \ldots, j\text{top}(i),\ i = 1, \ldots, m,$$

where the summation is over the index set $\{j \mid k - j \geq 0, j\text{bot}(i-1) \leq j \leq j\text{top}(i-1)\}$. Then $P = W(n+1, m)\,n!$.

Because machines with small exponent range on floating numbers cannot accommodate the array $W(\cdot, i)$ we have split the array into blocks. Each block contains the values of $W(\cdot, i)$ multiplied by a suitably large conversion factor. The reciprocals of these conversion factors are accumulated in TF by subroutine ACCUM; to keep the accumulated product TF within range it is multiplied by consecutive positive integers, as necessary, with the current value stored in NNN. At the conclusion of the program the value of TF is adjusted so that $NNN = N$ and then the multiplication by $n!$ in the formula $P = W(n+1, m)n!$ is accomplished by a multiplication by TF.

Subroutine DURB. Durbin's Method

The recursion is on $k!\,\alpha(k)$, $k!\,\beta(k)$ where α, β are defined in [1, eqs. (36), (37), (40)]. These variables are denoted by $ALPHA(k-1)$, $BETA(k-1)$ in our program, and the indexing runs through $k = 1, \ldots, n+1$. Durbin's algorithm computes the probability that $G_n(y)$ crosses a boundary so a subtraction from 1 is necessary. This instruction is flagged in the program listing.

Subroutine EPST. The Epanechnikov, Steck Method

This method [2, p. 15] is numerically unstable for larger values of n because of cancellation of digits in subtractions. Thus we do not include a listing of the computer program. However, we do include the test results for our implementation of this algorithm.

Organization of Programs

All three methods accept as inputs the boundary level points $y(k, j)$ and the sample size n. A preliminary computation of these boundary level points is necessary, and there must be $n+1$ of each. If $A(y) > (j-1)/n$ for all $y \in [0, 1]$, then a negative number must be given for $y(2, j)$ and if $B(y) < (j-1)/n$ for all $y \in [0, 1]$, then a number greater than 1 must be used for $y(1, j)$. As an example, we have included a listing of function PLN which computes P where A, B are (arbitrary) straight lines. As listed, PLN invokes method RAKK; the listing indicates the change necessary to invoke DURB or EPST.

Underflow occurs frequently in our programs and the underflow switch, if any, on the object machine should be set to return zero and the underflow diagnostic should be suppressed.

Test Results, Execution Times, and Precision

The three algorithms were programmed with single precision instructions only and were tested on the following machines: (a) DEC 10, (b) IBM 360-50, (c) CDC 6400. Single precision for these machines is, respectively, 8, 6, and 14 decimal digits and the range on decimal exponents is, respectively, -38 to 38, -75 to 75, and -293 to 321. The fortran compilers used were, respectively, FORTRAN 10/OPT, FORTRAN H(OPT = 2), and FUN. Table I lists the problems tested, where $A(y) = y + D$ and $B(y) = y - D$, and Table II gives the test results. If data is missing, the sample size n is too large for the machine and algorithm involved.

Table I. Problems Tested

Problem	n	D	P (approximate)
1	45	.0527	.001
2	200	.0257	.001
3	45	.1199	.500
4	120	.0742	.500
5	200	.0577	.500
6	45	.2849	.999
7	200	.1368	.999

Table II. Apparent Precision (in significant decimal digits) of Algorithms RAKK, DURB, and EPST

	RAKK			DURB			EPST		
	Machine			Machine			Machine		
Problem	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
1	7	5	12	4	2	9	7	5	12
2	6	4	11	—	2	9	4	—	10
3	6	4	12	6	5	12	3	2	10
4	6	3	11	—	4	12	—	—	7
5	6	3	11	—	4	11	—	—	4
6	6	3	11	8	6	12	—	—	3
7	6	2	10	2	5	12	—	—	—

Table III. Execution Times (in seconds) of Algorithms RAKK, DURB, and EPST

	RAKK			DURB			EPST		
	Machine			Machine			Machine		
Problem	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
1	.15	.53	.14	.56	3.08	.48	.03	.12	.02
2	1.73	5.53	1.44	—	71.10	11.21	.30	—	.24
3	.30	1.05	.29	.49	2.80	.41	.05	.32	.05
4	1.77	4.98	1.38	—	22.72	3.51	—	—	.24
5	4.53	11.40	3.22	—	64.52	10.54	—	—	.56
6	.73	1.80	.58	.38	1.92	.29	—	—	.10
7	10.54	28.72	7.24	19.62	53.70	8.84	—	—	—

The precision estimates for machines (a) and (b) are the number of digits which agree with the results of machine (c). The estimates for machine (c) were obtained by comparing the answers for the various methods. Since DURB computes the probability of the complementary event, the final subtraction from 1 accounts for the lesser precision of DURB in problems 1 and 2 and its greater precision in problems 6 and 7.

Execution times for arbitrary boundaries should be comparable to those given in Table III, plus the additional time needed to compute the level points $y(k, j)$.

Recommendations

Although EPST is numerically unstable for larger sample sizes, its execution times are very fast. Thus it would be suitable for numerous computations involving small sample sizes with occasional validations by one of the other routines. DURB is not suitable for large sample sizes on machines with small range on decimal exponents. Execution times for RAKK are faster. Thus RAKK appears to be the preferred choice. This is due largely to the extensive code with its controls against underflow, etc., rather than an intrinsic superiority of the algorithm.

Certification of Algorithm 487

Since Algorithm 487 [4], PKS2, treats a special case handled by Algorithms RAKK, DURB, and EPST, it may be validated by using the data presented in

Table IV. Apparent Precision (in significant decimal digits) of Algorithm PKS2

Machine	Problem Number 1	2	3	4	5	6
(a)	—	—	—	—	—	—
(b)	5	—	5	—	—	5
(c)	13+	—	12+	12+	—	13+

Table II. The results for the problems discussed are presented in Table IV. If data are missing the sample size is too large for the machine involved. (A computation of n^n is required by PKS2). Although Pomeranz claims precision of 8 decimal digits on the CDC 6400 [4], this seems rather modest. There is, apparently, a precision of at least 12 decimal digits on the CDC 6400 and a potentially greater precision if the double precision value generated by PKS2 is not returned as a single precision value.

ACKNOWLEDGMENT

These programs are an outgrowth of a seminar in computational probability given during 1974–1975 by Marcel Neuts of Purdue University, who also arranged for computer time on the Purdue machines.

REFERENCES

1. Durbin, J. Boundary crossing probabilities for the Brownian motion and Poisson processes and techniques for computing the power of the Kolmogrorov-Smirnov test. *J. Appl. Probability 8* (1971), 431–453.
2. Durbin, J. *Distribution Theory for Tests Based on the Sample Distribution Function.* SIAM, Philadelphia, Pa., 1973.
3. Massey, F. A note on the estimation of a distribution function by confidence limits. *Annals Math. Statist. 21* (1950), 116–119.
4. Pomeranz, J. Algorithm 487. Exact cumulative distribution of the Kolmogorov-Smirnov statistic for small samples. *Comm. ACM 17*, 12 (Dec. 1974), 703.
5. Wilks, S. *Mathematical Statistics.* Wiley, New York, 1962.

ALGORITHM

```
      FUNCTION PLN(N, S2, B2, S1, B1)                                  PLN   1Ø
C COMPUTES PROB FOR BOUNDARIES OF FORM S*Y+B.  N IS                     PLN   2Ø
C SAMPLE SIZE, S2,B2 ARE PARAMETERS FOR UPPER BOUNDARY,                 PLN   3Ø
C CALLS SUBROUTINE RAKK.  TO CALL DURB                                  PLN   4Ø
C CHANGE AS INDICATED BELOW.                                            PLN   5Ø
      DIMENSION YY(2,2Ø2), S(2), B(2)                                   PLN   6Ø
C MAX SIZE FOR N IS 2ØØ                                                 PLN   7Ø
      S(1) = S1                                                         PLN   8Ø
      S(2) = S2                                                         PLN   9Ø
      B(1) = B1                                                         PLN  1ØØ
      B(2) = B2                                                         PLN  11Ø
      IF (N.GT.Ø) GO TO 1Ø                                              PLN  12Ø
      PLN = Ø.                                                          PLN  13Ø
      RETURN                                                            PLN  14Ø
   1Ø IF (S(1).GE.Ø.) GO TO 2Ø                                          PLN  15Ø
      S(1) = Ø.                                                         PLN  16Ø
   2Ø IF (S(2).GE.Ø.) GO TO 3Ø                                          PLN  17Ø
      B(2) = S(2) + B(2)                                                PLN  18Ø
      S(2) = Ø.                                                         PLN  19Ø
   3Ø N1 = 1 + N                                                        PLN  2ØØ
      AN = N                                                            PLN  21Ø
      DO 5Ø I=1,2                                                       PLN  22Ø
        DO 4Ø K=1,N1                                                    PLN  23Ø
          C = FLOAT(K-1)/AN - B(I)                                      PLN  24Ø
          YY(I,K) = -.5                                                 PLN  25Ø
          IF (C.LE.Ø.) GO TO 4Ø                                         PLN  26Ø
          YY(I,K) = 1.1                                                 PLN  27Ø
          IF (S(I).GT.C) YY(I,K) = C/S(I)                               PLN  28Ø
   4Ø   CONTINUE                                                        PLN  29Ø
   5Ø CONTINUE                                                          PLN  3ØØ
      CALL RAKK(N, YY, ANS)                                             PLN  31Ø
```

```
C TO CALL DURB CHANGE PRECEDING STATEMENT TO                          PLN  32Ø
C    CALL DURB (N,YY,ANS)                                             PLN  33Ø
      PLN = ANS                                                       PLN  34Ø
      RETURN                                                          PLN  35Ø
      END                                                             PLN  36Ø


      SUBROUTINE RAKK(N, YY, ANS)                                     RAK   1Ø
C N IS SAMPLE SIZE.  YY(2,.),YY(1,.)                                  RAK   2Ø
C CONTAIN, RESPECTIVELY, UPPER AND LOWER BOUNDARY                     RAK   3Ø
C LEVEL POINTS CORRESPONDING TO Ø,1/N,2/N,...,1.  MUST                RAK   4Ø
C BE N+1 OF EACH.  ROUTINE ASSUMES THEY ARE NONDECREASING.            RAK   5Ø
      DIMENSION W(2Ø1,2), YY(2,2Ø2), KBLOC(41), FRNG(41,2), SSS(41)   RAK   6Ø
      DATA RNG, RRNG, SQRNG, SQRRNG, MAXNB /1.E292,1.E-292,1.E146,    RAK   7Ø
     * 1.E-146,4Ø/                                                    RAK   8Ø
C RNG AND RRNG ARE MACHINE DEPENDENT CONSTANTS SELECTED               RAK   9Ø
C SO RRNG=1./RNG G.E. SMALLEST POSITIVE FLOATING                      RAK  1ØØ
C NUMBER AND RNG L.E. LARGEST POSITIVE FLOATING                       RAK  11Ø
C NUMBER OF MACHINE. SQRNG AND SQRRNG ARE SQUARE ROOTS                RAK  12Ø
C OF RNG AND RRNG.                                                    RAK  13Ø
C MAXNB IS THE MAX NUMBER OF BLOCKS INTO WHICH                        RAK  14Ø
C W(.,.) WILL BE PARTITIONED.  IF MAXNB IS                            RAK  15Ø
C INCREASED REVISE DIMENSION STATEMENT ACCORDINGLY.                   RAK  16Ø
      N1 = N + 1                                                      RAK  17Ø
      TF = 1.                                                         RAK  18Ø
      NNN = 1                                                         RAK  19Ø
      IEXT = 1                                                        RAK  2ØØ
      M2 = 1                                                          RAK  21Ø
      M1 = 2                                                          RAK  22Ø
      W(1,M2) = SQRNG                                                 RAK  23Ø
      NBLOC = 1                                                       RAK  24Ø
      KBLOC(1) = 1                                                    RAK  25Ø
      KBLOC(2) = 2                                                    RAK  26Ø
      FRNG(1,M2) = 1.                                                 RAK  27Ø
      ANS = Ø.                                                        RAK  28Ø
      IF (N1.LE.1) RETURN                                             RAK  29Ø
      YY(2,N1+1) = 1.                                                 RAK  3ØØ
      Y1 = Ø.                                                         RAK  31Ø
      IF ((YY(1,1).LE.Ø.) .OR. (YY(2,N1).GE.1.) .OR.                  RAK  32Ø
     * (YY(1,N1).LE.1.) .OR. (YY(2,1).GE.Ø.)) RETURN                  RAK  33Ø
      JBOTP = 1                                                       RAK  34Ø
      JTOPP = 1                                                       RAK  35Ø
      JTOP1 = 1                                                       RAK  36Ø
C ENTRY POINT FOR SUBSEQUENT ITERATIONS.                              RAK  37Ø
   1Ø JBOTØ = JBOTP                                                   RAK  38Ø
      JTOPØ = JTOP1                                                   RAK  39Ø
      MT = M1                                                         RAK  4ØØ
      M1 = M2                                                         RAK  41Ø
      M2 = MT                                                         RAK  42Ø
      YØ = Y1                                                         RAK  43Ø
C COMPUTE SUMMATION INDICES, JBOTP,JTOPP FOR NEXT                     RAK  44Ø
C ITERATION.                                                          RAK  45Ø
   2Ø IF (YY(2,JTOPP+1).GT.YØ) GO TO 3Ø                               RAK  46Ø
      JTOPP = JTOPP + 1                                               RAK  47Ø
      GO TO 2Ø                                                        RAK  48Ø
   3Ø Y1 = AMIN1(YY(2,JTOPP+1),YY(1,JBOTP))                           RAK  49Ø
      IF (Y1.GE.1.) GO TO 5Ø                                          RAK  5ØØ
   4Ø IF (YY(1,JBOTP).GT.Y1) GO TO 6Ø                                 RAK  51Ø
      JBOTP = JBOTP + 1                                               RAK  52Ø
      GO TO 4Ø                                                        RAK  53Ø
   5Ø Y1 = 1.                                                         RAK  54Ø
C SET EXIT FLAG.  NEXT ITERATION IS FINAL ONE.                        RAK  55Ø
      IEXT = 2                                                        RAK  56Ø
      JBOTP = N1                                                      RAK  57Ø
   6Ø IF (JBOTP.GT.JTOPP) RETURN                                      RAK  58Ø
C RETURN WITH ANS=Ø SINCE NO PATHS BETWEEN BOUNDARIES.                RAK  59Ø
      JTOP1 = JTOPP                                                   RAK  6ØØ
      P = Y1 - YØ                                                     RAK  61Ø
      KBLOC(NBLOC+1) = JTOP1 + 1                                      RAK  62Ø
      DO 7Ø L=JBOTØ,JTOP1                                             RAK  63Ø
        W(L,M2) = Ø.                                                  RAK  64Ø
   7Ø CONTINUE                                                        RAK  65Ø
      LØ = KBLOC(NBLOC)                                               RAK  66Ø
      L1 = JTOPØ                                                      RAK  67Ø
      DO 8Ø L=LØ,L1                                                   RAK  68Ø
```

```
            IF (W(L,M1).GT.SQRRNG) GO TO 8Ø                          RAK  69Ø
            JTOPØ = L - 1                                            RAK  7ØØ
            GO TO 9Ø                                                 RAK  71Ø
       8Ø CONTINUE                                                   RAK  72Ø
       9Ø IF (JBOTØ.LT.KBLOC(2)) GO TO 11Ø                           RAK  73Ø
C THE NEXT FEW STATEMENTS ELIMINATE BLOCKS NO                        RAK  74Ø
C LONGER USED.                                                       RAK  75Ø
          DO 1ØØ I=1,NBLOC                                           RAK  76Ø
            FRNG(I,M1) = FRNG(I+1,M1)                                RAK  77Ø
            KBLOC(I) = KBLOC(I+1)                                    RAK  78Ø
     1ØØ CONTINUE                                                    RAK  79Ø
          NBLOC = NBLOC - 1                                          RAK  8ØØ
          CALL ACCUM(RRNG/FRNG(1,M1), TF, NNN)                       RAK  81Ø
          GO TO 9Ø                                                   RAK  82Ø
     11Ø KBLOC(1) = JBOTØ                                            RAK  83Ø
          MBAV = (MAXNB-NBLOC+1)/2                                   RAK  84Ø
          IF (P.LE.(1./FLOAT(4*N1))) MBAV = 2                        RAK  85Ø
          MBAV = MINØ(MAXNB,NBLOC+MBAV)                              RAK  86Ø
C MBAV IS UPPER LIMIT ON NUMBER OF BLOCKS DURING NEXT                RAK  87Ø
C ITERATION.  MBAV IS SET TO LIMIT NUMBER OF NEW                     RAK  88Ø
C BLOCKS WHEN P IS SMALL.                                            RAK  89Ø
          CONVI = 1.                                                 RAK  9ØØ
          I1 = 1                                                     RAK  91Ø
          J1 = 1                                                     RAK  92Ø
     12Ø DO 4ØØ I=I1,NBLOC                                           RAK  93Ø
            KBLI = KBLOC(I)                                          RAK  94Ø
            IF (I.EQ.I1) GO TO 17Ø                                   RAK  95Ø
            FRNG(I,M2) = AMIN1(FRNG(I,M1),1./(SQRNG*W(KBLI-1,M2)))   RAK  96Ø
            CONVI = (CONVI/FRNG(I-J1+1,M1))*FRNG(I,M2)               RAK  97Ø
            IF (J1.EQ.1) GO TO 17Ø                                   RAK  98Ø
            K1 = KBLOC(I) - KBLOC(I-J1+2) + 1                        RAK  99Ø
            KØ = KBLOC(I-1) - KBLOC(I-J1+1) + 1                      RAK 1ØØØ
            IF (K1-KØ) 15Ø, 17Ø, 13Ø                                 RAK 1Ø1Ø
     13Ø    KØP1 = KØ + 1                                            RAK 1Ø2Ø
            DO 14Ø K=KØP1,K1                                         RAK 1Ø3Ø
              CONVI = CONVI*P/FLOAT(K)                               RAK 1Ø4Ø
     14Ø    CONTINUE                                                 RAK 1Ø5Ø
            GO TO 17Ø                                                RAK 1Ø6Ø
     15Ø    K1P1 = K1 + 1                                            RAK 1Ø7Ø
            DO 16Ø K=K1P1,KØ                                         RAK 1Ø8Ø
              CONVI = CONVI*FLOAT(K)/P                               RAK 1Ø9Ø
     16Ø    CONTINUE                                                 RAK 11ØØ
     17Ø    IF (CONVI.GT.RRNG) GO TO 21Ø                             RAK 111Ø
            DO 19Ø J=J1,J11                                          RAK 112Ø
C CHANGE UPPER LIMIT IF EARLY EXIT FROM PREVIOUS LOOP                RAK 113Ø
              CONVI = FRNG(I,M2)*RNG                                 RAK 114Ø
              KØ = MAXØ(1,KBLOC(I-1)-MINØ(JTOPØ+1,KBLOC(I-J+1))+2)   RAK 115Ø
              K1 = KBLOC(I) - MINØ(JTOPØ+1,KBLOC(I-J+1)) + 1         RAK 116Ø
              DO 18Ø K=KØ,K1                                         RAK 117Ø
                CONVI = CONVI*P/FLOAT(K)                             RAK 118Ø
     18Ø      CONTINUE                                               RAK 119Ø
              CONVI = SSS(J)*CONVI                                   RAK 12ØØ
              IF (CONVI.GT.RRNG) GO TO 2ØØ                           RAK 121Ø
     19Ø    CONTINUE                                                 RAK 122Ø
            GO TO (1Ø, 44Ø), IEXT                                    RAK 123Ø
     2ØØ    J1 = J + 1                                               RAK 124Ø
     21Ø    DO 39Ø J=J1,I                                            RAK 125Ø
              LFG = 1                                                RAK 126Ø
C LFG=2 IF NEXT ITERATION ON J BECOMES NECESSARY.                    RAK 127Ø
              LB1 = KBLOC(I)                                         RAK 128Ø
              KØ1 = KBLOC(I) - KBLOC(I-J+1) + 1                      RAK 129Ø
              S = CONVI                                              RAK 13ØØ
              IF (J.EQ.J1) GO TO 24Ø                                 RAK 131Ø
              IF (TMLT) 22Ø, 4ØØ, 23Ø                                RAK 132Ø
     22Ø      S = -TMLT*FRNG(I-J+2,M1)                               RAK 133Ø
              IF (S.GT.RRNG) GO TO 24Ø                               RAK 134Ø
     23Ø      S = TMLT*(FRNG(I-J+2,M1)*RNG)                          RAK 135Ø
     24Ø      KØ = MAXØ(KBLOC(I)-MINØ(KBLOC(I-J+2)-1,JTOPØ)+1,1)     RAK 136Ø
              K1 = KBLOC(I+1) - KBLOC(I-J+1)                         RAK 137Ø
              TMLT = Ø.                                              RAK 138Ø
              RRNGS = 1.                                             RAK 139Ø
              IF (S.LT.Ø.) RRNGS = -RRNG                             RAK 14ØØ
              SSS(J) = S                                             RAK 141Ø
              DO 37Ø K=KØ,K1                                         RAK 142Ø
                LDF = 1 - K                                          RAK 143Ø
```

```
            LB = MAXØ(KBLOC(I),KBLOC(I-J+1)-LDF,LB1)                    RAK 144Ø
            LT = MINØ(KBLOC(I+1)-1,MINØ(JTOPØ,KBLOC(I-J+2)-1)-LDF)      RAK 145Ø
            IF (LB.GT.LB1) LFG = 2                                      RAK 146Ø
            IF (LB.GT.LT) GO TO 27Ø                                     RAK 147Ø
            DO 26Ø L=LB,LT                                              RAK 148Ø
              IF (W(L,M2).EQ.((W(L+LDF,M1)*RRNGS)*S+W(L,M2))) GO TO     RAK 149Ø
     *         25Ø                                                      RAK 15ØØ
              LB1 = L                                                   RAK 151Ø
C WHEN THE SUMMANDS BECOME SO SMALL THAT THE VALUE OF                   RAK 152Ø
C W(L,M2) IS UNCHANGED LB1 IS INCREASED.  THIS PREVENTS                 RAK 153Ø
C FURTHER ADDITIONS TO SUCH W(L,M2).                                    RAK 154Ø
            GO TO 29Ø                                                   RAK 155Ø
  25Ø       IF (W(L,M2).GT.RRNG) GO TO 26Ø                              RAK 156Ø
            IF (K-KØ1) 34Ø, 34Ø, 28Ø                                    RAK 157Ø
  26Ø       CONTINUE                                                    RAK 158Ø
  27Ø       LB1 = LT + 1                                                RAK 159Ø
            IF ((LB1.NE.KBLOC(I+1)) .OR. (K.LE.KØ1)) GO TO 34Ø          RAK 16ØØ
  28Ø       J11 = J                                                     RAK 161Ø
            IF (W(KBLI,M2).LE.RRNG) LFG = 2                             RAK 162Ø
            GO TO (4ØØ, 38Ø), LFG                                       RAK 163Ø
  29Ø       IF (S) 3ØØ, 38Ø, 32Ø                                        RAK 164Ø
  3ØØ       DO 31Ø L=LB1,LT                                             RAK 165Ø
              W(L,M2) = (W(L+LDF,M1)*RRNGS)*S + W(L,M2)                 RAK 166Ø
  31Ø       CONTINUE                                                    RAK 167Ø
            GO TO 34Ø                                                   RAK 168Ø
  32Ø       DO 33Ø L=LB1,LT                                             RAK 169Ø
              W(L,M2) = W(L+LDF,M1)*S + W(L,M2)                         RAK 17ØØ
  33Ø       CONTINUE                                                    RAK 171Ø
  34Ø       T = S                                                       RAK 172Ø
            S = S*P/FLOAT(K)                                            RAK 173Ø
            IF (S) 36Ø, 35Ø, 36Ø                                        RAK 174Ø
  35Ø       IF (T.LE.Ø.) GO TO 38Ø                                      RAK 175Ø
            S = -(T*RNG)*P/FLOAT(K)                                     RAK 176Ø
            RRNGS = -RRNG                                               RAK 177Ø
C THE NEGATIVE BIT IN S,TMLT IS USED TO                                 RAK 178Ø
C INDICATE ITS VALUE IS RNG TIMES THE USUAL VALUE.                      RAK 179Ø
  36Ø       IF (K.EQ.KØ1) TMLT = S                                      RAK 18ØØ
C THE ITERATION ON K MUST CONTINUE UNTIL TMLT IS                        RAK 181Ø
C SET.  TMLT USED IN NEXT ITERATION ON J.                               RAK 182Ø
  37Ø     CONTINUE                                                      RAK 183Ø
  38Ø     J11 = J                                                       RAK 184Ø
  39Ø   CONTINUE                                                        RAK 185Ø
  4ØØ CONTINUE                                                          RAK 186Ø
C CREATE NEW BLOCK IF NECESSARY.                                        RAK 187Ø
      IF (NBLOC.GE.MBAV) GO TO (1Ø, 44Ø), IEXT                          RAK 188Ø
      L1 = KBLOC(NBLOC+1) - KBLOC(NBLOC)                                RAK 189Ø
      DO 41Ø L=1,L1                                                     RAK 19ØØ
        LL = KBLOC(NBLOC+1) - L                                         RAK 191Ø
        IF (W(LL,M2).GE.SQRRNG) GO TO 42Ø                               RAK 192Ø
  41Ø CONTINUE                                                          RAK 193Ø
  42Ø IF (LL.EQ.JTOP1) GO TO (1Ø, 44Ø), IEXT                            RAK 194Ø
      KBLOC(NBLOC+2) = KBLOC(NBLOC+1)                                   RAK 195Ø
      KBLOC(NBLOC+1) = LL + 1                                           RAK 196Ø
      NBLOC = NBLOC + 1                                                 RAK 197Ø
      LØ = KBLOC(NBLOC)                                                 RAK 198Ø
      DO 43Ø L=LØ,JTOP1                                                 RAK 199Ø
        W(L,M2) = Ø.                                                    RAK 2ØØØ
  43Ø CONTINUE                                                          RAK 2Ø1Ø
      I1 = NBLOC                                                        RAK 2Ø2Ø
      FRNG(NBLOC,M2) = 1.                                               RAK 2Ø3Ø
      CONVI = Ø.                                                        RAK 2Ø4Ø
      GO TO 12Ø                                                         RAK 2Ø5Ø
  44Ø CALL ACCUM(W(N1,M2), TF, NNN)                                     RAK 2Ø6Ø
      CALL ACCUM(SQRRNG, TF, NNN)                                       RAK 2Ø7Ø
      IF (NBLOC.EQ.1) GO TO 46Ø                                         RAK 2Ø8Ø
      DO 45Ø I=2,NBLOC                                                  RAK 2Ø9Ø
        CALL ACCUM(RRNG/FRNG(I,M2), TF, NNN)                            RAK 21ØØ
  45Ø CONTINUE                                                          RAK 211Ø
  46Ø ANS = TF                                                          RAK 212Ø
      IF (N-NNN) 47Ø, 49Ø, 5ØØ                                         RAK 213Ø
  47Ø DO 48Ø J=N1,NNN                                                   RAK 214Ø
        ANS = ANS/FLOAT(J)                                              RAK 215Ø
  48Ø CONTINUE                                                          RAK 216Ø
  49Ø RETURN                                                            RAK 217Ø
  5ØØ NNN = NNN + 1                                                     RAK 218Ø
```

```
      DO 51Ø J=NNN,N                                                    RAK 219Ø
        ANS = ANS*FLOAT(J)                                              RAK 22ØØ
  51Ø CONTINUE                                                          RAK 221Ø
      RETURN                                                            RAK 222Ø
      END                                                               RAK 223Ø


      SUBROUTINE ACCUM(F, TF, NNN)                                      ACC   1Ø
C THIS SUBROUTINE ANCILLARY TO RAKK AND ACCUMULATES THE                 ACC   2Ø
C MULTIPLIERS USED TO KEEP W(.,.) IN RANGE.                             ACC   3Ø
      TF = TF*F                                                         ACC   4Ø
   1Ø IF ((TF.GT.1.) .OR. (TF.EQ.Ø.)) RETURN                            ACC   5Ø
      NNN = NNN + 1                                                     ACC   6Ø
      TF = TF*FLOAT(NNN)                                                ACC   7Ø
      GO TO 1Ø                                                          ACC   8Ø
      END                                                               ACC   9Ø


      SUBROUTINE DURB(N, YY, ANS)                                       DUR   1Ø
C DURBINS METHOD.  J.APPL.PROB., 8(1971), PP431-453,                    DUR   2Ø
C FORMULAS (36),(37),(4Ø).                                              DUR   3Ø
C N IS SAMPLE SIZE.  YY(2,.),YY(1,.)                                    DUR   4Ø
C CONTAIN, RESPECTIVELY, UPPER AND LOWER BOUNDARY                       DUR   5Ø
C LEVEL POINTS CORRESPONDING TO Ø,1/N,2/N,...,1.  MUST                  DUR   6Ø
C BE N+1 OF EACH.  ROUTINE ASSUMES THEY ARE NONDECREASING.              DUR   7Ø
      DIMENSION ALPHA(2Ø1), BETA(2Ø1), YY(2,2Ø2)                        DUR   8Ø
C DIMENSION STATEMENT PERMITS MAX SAMPLE SIZE OF 2ØØ                    DUR   9Ø
      N1 = N + 1                                                        DUR  1ØØ
      ANS = Ø.                                                          DUR  11Ø
      IF ((YY(1,1).LE.Ø.) .OR. (YY(2,N1).GE.1.) .OR.                    DUR  12Ø
     * (YY(1,N1).LE.1.) .OR. (YY(2,1).GE.Ø.)) RETURN                    DUR  13Ø
      IP = 1                                                            DUR  14Ø
      DO 2Ø I=1,N1                                                      DUR  15Ø
        IF (YY(2,I).LT.Ø.) GO TO 1Ø                                     DUR  16Ø
        IP = I                                                          DUR  17Ø
        GO TO 3Ø                                                        DUR  18Ø
   1Ø   YY(2,I) = Ø.                                                    DUR  19Ø
   2Ø CONTINUE                                                          DUR  2ØØ
   3Ø DO 4Ø I=1,N1                                                      DUR  21Ø
        IF (YY(1,I).LE.1.) GO TO 4Ø                                     DUR  22Ø
        YY(1,I) = 1.                                                    DUR  23Ø
   4Ø CONTINUE                                                          DUR  24Ø
      ALPHA(1) = Ø.                                                     DUR  25Ø
      BETA(1) = 1.                                                      DUR  26Ø
      DO 12Ø K=2,N1                                                     DUR  27Ø
        ALPHA(K) = YY(2,K)**(K-1)                                       DUR  28Ø
        IF (IP.GT.(K-1)) GO TO 6Ø                                       DUR  29Ø
        COMB = 1.                                                       DUR  3ØØ
        I1 = K - IP                                                     DUR  31Ø
        DO 5Ø I=1,I1                                                    DUR  32Ø
          COMB = COMB*FLOAT(K-I)/FLOAT(I)                               DUR  33Ø
C COMB IS BINOMIAL COEFFICIENT K-1 OVER I-1.                            DUR  34Ø
          ALPHA(K) = ALPHA(K) - ALPHA(K-I)*(YY(2,K)-YY(2,K-I))**I*      DUR  35Ø
     *     COMB                                                         DUR  36Ø
   5Ø   CONTINUE                                                        DUR  37Ø
   6Ø   COMB = 1.                                                       DUR  38Ø
        DO 7Ø J=1,N1                                                    DUR  39Ø
          IF (YY(2,K).LE.YY(1,J)) GO TO 8Ø                              DUR  4ØØ
          ALPHA(K) = ALPHA(K) - BETA(J)*(YY(2,K)-YY(1,J))**(K-J)*       DUR  41Ø
     *     COMB                                                         DUR  42Ø
C COMB IS BINOMIAL COEFFICIENT K-1 OVER J-1.                            DUR  43Ø
          COMB = COMB*FLOAT(K-J)/FLOAT(J)                               DUR  44Ø
   7Ø   CONTINUE                                                        DUR  45Ø
   8Ø   BETA(K) = YY(1,K)**(K-1)                                        DUR  46Ø
        IF (IP.GT.K) GO TO 1ØØ                                          DUR  47Ø
        COMB = 1.                                                       DUR  48Ø
        I1 = K - IP + 1                                                 DUR  49Ø
        DO 9Ø I=1,I1                                                    DUR  5ØØ
          BETA(K) = BETA(K) - ALPHA(K-I+1)*(YY(1,K)-YY(2,K-I+1))**      DUR  51Ø
     *     (I-1)*COMB                                                   DUR  52Ø
C COMB IS BINOMIAL COEFFICIENT K-1 OVER I-1.                            DUR  53Ø
          COMB = COMB*FLOAT(K-I)/FLOAT(I)                               DUR  54Ø
   9Ø   CONTINUE                                                        DUR  55Ø
        COMB = 1.                                                       DUR  56Ø
```

```
  1ØØ   KM1 = K - 1                                                      DUR  57Ø
        DO 11Ø J=1,KM1                                                   DUR  58Ø
          IF (YY(1,J).EQ.1.) GO TO 12Ø                                   DUR  59Ø
C COMB IS BINOMIAL COEFFICIENT K-1 OVER J-1.                             DUR  6ØØ
          BETA(K) = BETA(K) - BETA(J)*(YY(1,K)-YY(1,J))**(K-J)*COMB      DUR  61Ø
          COMB = COMB*FLOAT(K-J)/FLOAT(J)                                DUR  62Ø
  11Ø   CONTINUE                                                         DUR  63Ø
  12Ø CONTINUE                                                           DUR  64Ø
      COMB = 1.                                                          DUR  65Ø
      I1 = N1 - IP + 1                                                   DUR  66Ø
      DO 13Ø I=1,I1                                                      DUR  67Ø
        ANS = ANS + (1.-YY(2,N1-I+1))**(I-1)*ALPHA(N1-I+1)*COMB          DUR  68Ø
        COMB = COMB*FLOAT(N1-I)/FLOAT(I)                                 DUR  69Ø
  13Ø CONTINUE                                                           DUR  7ØØ
      COMB = 1.                                                          DUR  71Ø
      DO 14Ø J=1,N1                                                      DUR  72Ø
        IF (YY(1,J).EQ.1.) GO TO 15Ø                                     DUR  73Ø
        ANS = ANS + (1.-YY(1,J))**(N1-J)*BETA(J)*COMB                    DUR  74Ø
C COMB IS BINOMIAL COEFFICIENT N OVER J-1.                               DUR  75Ø
        COMB = COMB*FLOAT(N1-J)/FLOAT(J)                                 DUR  76Ø
  14Ø CONTINUE                                                           DUR  77Ø
C FOR PROB OF COMPLEMENTARY EVENT CHANGE NEXT STATEMENT TO               DUR  78Ø
C 15Ø CONTINUE                                                           DUR  79Ø
  15Ø ANS = 1. - ANS                                                     DUR  8ØØ
      RETURN                                                             DUR  81Ø
      END                                                                DUR  82Ø
```

ALGORITHM 520
An Automatic Revised Simplex Method for Constrained Resource Network Scheduling [H]

JAN WEGLARZ, JACEK BLAZEWICZ, WOJCIECH CELLARY, and ROMAN SLOWINSKI

Technical University of Poznan, Poland

Key Words and Phrases: resource allocation, activity networks, linear programming
CR Categories: 3.31, 4.32, 5.41, 8.3
Language: Fortran

DESCRIPTION

Purpose

Subroutine ARSME solves a resource constrained, network scheduling problem for the case in which activities may be arbitrarily interrupted and restarted later with no increase in activity duration. The number of resource types is not a limiting factor in our procedure. The amount of any one resource available at any moment is constant. We shall use the "activity-on-arc" network representation, under the commonly imposed assumption that the network contains no directed cycles and has only one "beginning" and only one "terminal" node (event). It is further assumed that the network nodes (events) are ordered in such a way that node i precedes node j, if $i < j$. Such an ordering is always possible and it induces an ordering among the arcs (activities).

Optimal approaches to resource constrained, network scheduling problems where activities can require more than one resource type are presented in [1, 4]. Both methods assume integer durations of activities, and the method presented in [1] divides activity durations into unit intervals. Both methods can handle networks with up to about 30 activities and 3 resource types.

Subroutine ARSME is constructed in such a way that its storage requirements are minimal, a fact which permits the solution of problems for very large networks with many resource types. Moreover, an optimal solution can be obtained in a shorter time when relatively smaller amounts of the resources are available than when resources are less limited.

Let the number of activities be equal to M and the number of resource types be equal to RT. For activity j ($j = 1, 2, \ldots, M$) and resource k ($k = 1, 2, \ldots, RT$) the following variables are given: activity duration d_j; activity resource requirement R_{jk}, i.e. the amount of resource k required by activity j; resource limit RL_k, i.e. the amount of resource k available at any moment.

The solution of our problem consists of the assignment of time intervals to resource feasible sets of activities (i.e. activities which can be performed simul-

Received 18 December 1974, 5 June 1975, and 15 June 1976.

Authors' address: Institute of Control Engineering, Technical University of Poznan, Piotrowo 3a, 60 965 Poznan, Poland.

taneously and not violate resource limits), in order that the completion time of the project be minimal. For this reason, it is necessary to generate all such sets and to solve the ensuing linear programming (LP) representation of the problem. LP variables are the durations assigned to these activity sets. The optimal durations obtained are ordered according to the network structure. In similar fashion we obtain the time intervals making up the optimal schedule.

As input data for subroutine ARSME, it is necessary to provide the following data:

M	number of activities in the network;
$M1$	dimension limit equal to $M + 1$;
$M2$	dimension limit equal to $2M + 1$;
ME	dimension limit of vector E (see the discussion below);
ND	number of nodes;
RT	number of resource types;
NI	declared limit of the number of iterations, usually taken as $2M$ (the subroutine ends computations if the optimum solution is not found in NI iterations);
$NS(M2)$	network structure vector containing the consecutive activities as ordered pairs of nodes; no initial value is required for element $NS(M2)$;
$X(M1)$	vector of activity durations;
$RL(RT)$	vector of resource limits;
$R(M, RT)$	matrix of resource requirements.

The value of ME is taken to be $(M1 \times NI)$ if $M1 \times NI$ does not exceed the available amount of primary storage; otherwise, it is taken as the available limit of primary storage.

When the optimal solution is obtained, the optimal schedule (i.e. the sets of activities together with corresponding time intervals) are printed on the line printer. On the other hand, if the optimal solution is not reached in the declared number of iterations NI, then the message "number of iterations greater than NI" is printed. In this case, the computations should be repeated for a greater value of NI.

As an example let us consider the network shown in Figure 1. Input data are: $M = 5$; $M1 = 6$; $M2 = 11$; $ME = 60$; $ND = 4$; $RT = 3$; $NI = 10$; $NS = [1, 2, 1, 3, 2, 3, 2, 4, 3, 4]$; $X = [1.0, 2.0, 3.0, 2.0, 2.0]$; $RL = [5.0, 5.0, 3.0]$;

$$R = \begin{pmatrix} 2.0, & 2.0, & 1.0 \\ 0.0, & 2.0, & 1.0 \\ 2.0, & 1.0, & 3.0 \\ 3.0, & 3.0, & 3.0 \\ 1.0, & 1.0, & 0.0 \end{pmatrix}.$$

As an optimal schedule, ARSME yields

time intervals	activities				
	1	2	3	4	5
0.0–1.0	1	1	0	0	0
1.0–2.0	0	1	0	0	0
2.0–5.0	0	0	1	0	0
5.0–7.0	0	0	0	1	1
7.0–7.0	0	0	0	0	1

with $T_{\min} = 7$. This means that, in the optimal schedule, activities 1 and 2 should be performed simultaneously in the time interval $\langle 0.0, 1.0 \rangle$, activity 2 in the time interval $\langle 1.0, 2.0 \rangle$, etc.

Method

Subroutine ARSME solves an LP problem which is formulated as follows.

Let S_I denote the set of all activities which may be performed between the oc-

currence of node I and $I + 1$. Such sets will be called *main sets*. Let us number from 1 to N the resource feasible sets, i.e. those subsets of the main sets and those main sets for which the resource requirements do not exceed any resource limits.

Now, let Q_j denote the set of all numbers of resource feasible sets in which activity j may be performed, and let x_i denote the duration of set i. Thus the LP problem is obtained:

minimize

$$T = \sum_{i=1}^{N} x_i$$

subject to

$$\sum_{i \in Q_j} x_i = d_j \qquad (j = 1, 2, \ldots, M) \tag{1}$$

or in matrix notation,

$$\mathbf{A}\,\mathbf{x} = \mathbf{d} \tag{2}$$

where $\mathbf{A}$ is the matrix of coefficients

$$a_{ji} = \begin{cases} 1 & \text{if } i \in Q_j; \\ 0 & \text{otherwise.} \end{cases}$$

Obviously, columns of matrix $\mathbf{A}$ correspond to resource feasible sets.

When applying the simplex method directly to this problem, one must somehow store matrix $\mathbf{A}$, which is a very large matrix for even small problems. This greatly constrains the size of the problems which can be solved.

Subroutine ARSME eliminates the above storage problem by utilizing the revised simplex method [2] in which elements of matrix $\mathbf{A}$ are not transformed. This method is computationally profitable for sequencing problems, since in practical ones, the number of variables is greater than three times the number of constraints [5]. The specific version used is based upon the product form of the inverse [2].

The first basic feasible solution to the problem is known in advance. It consists of M, 1-element sets (consecutive performance of all activities).

Taking advantage of the special structure of the problem, a subroutine GEN has been developed which automatically generates the consecutive columns of matrix $\mathbf{A}$ (i.e. the resource feasible sets). An important feature of this subroutine is that at every moment only one main set and only one of its resource feasible subsets (which is actually needed in the simplex procedure) are stored. This allows one to solve problems of larger dimension than otherwise possible and of course radically simplifies the preparation of input data.

Subroutine GEN generates main sets $S_1, S_2, \ldots, S_{ND}$ using the network structure vector. When a single main set has been generated, the generation of its resource feasible subsets follows. For this purpose, so-called *primary subsets* are constructed, from which resource feasible subsets are created by adding activities from the main set. The first primary subset contains all the activities of the main set which also composed previously generated main sets, as well as one new activity. New primary subsets are created either when the last activity from the main set has been added, or when the generated subset is found to be unfeasible. In the first case, the last activity of the last generated subset is rejected and the penultimate one is replaced by the next from the main set. In the second case, the last activity of the unfeasible subset is replaced by the next from the main set. The number of sets in which resource feasibility is checked is always minimal and is usually much smaller than the total number of sets of activities which may be performed simultaneously. This follows from the construction of the primary subsets and from the fact that activities in the main sets are ordered by the subroutine ORDSNE in a unique fashion.

After generating each resource feasible set ARSME, together with subrou-

tine KFIND, checks, according to the simplex procedure, whether the set is good enough to be in the optimal solution. This procedure is repeated until the optimal solution is obtained. Then subroutine ORDSNE orders the sets composing the optimal solution, and subroutine PRISET prints out the results.

The parameters of ARSME, which are associated with the applied version of the simplex procedure, are as follows:

- *U, D* vector used for finding vectors introduced into, and eliminated from, the basis, respectively;
- *E* vector of the inverse;
- *LO* vector of the indices of vectors eliminated from the basis in successive iterations;
- *IBV* vector of indices of the basis vectors.

(Other parameters are subsidiary ones).

Timing Tests

A comparison of the computation time required by subroutine ARSME and other methods for solving the constrained resource network scheduling problem is not very helpful because of the different assumptions made in these methods. However, as an example, a comparison of computation times is shown in Table I for the two

Table I

Problem	Resource limits RL_k	Davis method		Patterson method		ARSME	
		Optimal schedule length	Time to optimal solution*	Optimal schedule length	Time to optimal solution‡	Optimal schedule length§	Time to optimal solution†
1. Davis and Heidorn	5, 5, 3	7		7	1.57	7	0.81
2. Moodie and Mandeville	5	8	2.00	8	1.35	8	1.20
3. Moodie and Mandeville	4	11		11	3.78	10.5	1.14

* IBM 7094 CPU time in seconds.
† IBM 360/67 CPU time in seconds.
‡ ICL 1900 CPU time in seconds.
§ Because of the provision for job splitting in our procedure, optimal schedule lengths may differ from those reported by Davis and by Patterson.

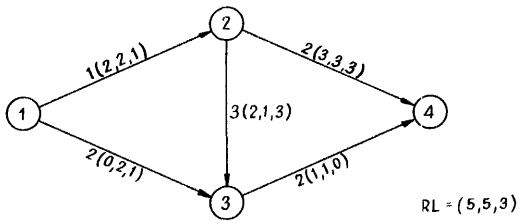


Fig. 1. Network used in problem formulated by Davis and Heidorn [1]

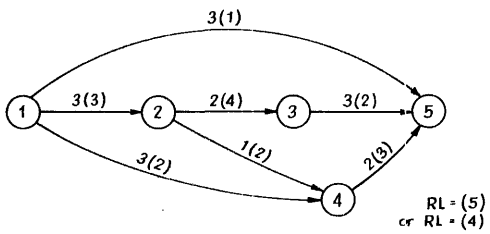


Fig. 2. Network used in problem formulated by Moodie and Mandeville [3]

simple networks described in [1, 3] and shown in Figures 1 and 2. One may notice that, for problem 3, ARSME finds a solution with a lower value of the objective function, because of the assumption of the arbitrary splitting of activities.

All tests have been made on an ODRA-1305 computer (license ICL-1900). Computation times for networks of about 20 activities vary from about 15 seconds to 3 minutes; for networks of about 50 activities, from 5 minutes to 25 minutes; and for networks of about 100 activities, from 20 minutes to 90 minutes. Variance in computation times for a given size of network appears to be primarily a function of the number of network nodes and the values of resource limits.

ACKNOWLEDGMENTS

The authors are indebted to Michael J. D. Powell and to James H. Patterson who made some valuable remarks for the overall improvement of this paper.

REFERENCES

1. DAVIS, E.W., AND HEIDORN, G.E. An algorithm for optimal project scheduling under multiple resource constraints. *Manage. Sci. 17*, 12 (Aug. 1971), B803–B816.
2. GASS, S.I. *Linear Programming*. McGraw-Hill, New York, 1969.
3. MOODIE, C.L., AND MANDEVILLE, D.E., Project resource balancing by assembly line balancing techniques. *J. Indust. Eng. 17*, 7 (July 1966), 377–383.
4. PATTERSON, J.H., AND HUBER, W.D., A horizon varying zero-one approach to project scheduling. *Manage. Sci. 20*, 6 (Feb. 1974), 900–998.
5. WAGNER, H.N. A comparison of the original and revised simplex methods. *Oper. Res. 5*, 3 (June 1957), 361–369.

ALGORITHM

[Only that portion of the listing which is the introductory comment section explaining subroutine ARSME is printed here. The complete listing, including subroutines GEN, KFIND, PRISET, and ORDSNE, is available from the ACM Algorithms Distribution Service (see inside back cover for order form), or may be found in "Collected Algorithms from ACM."]

```
      SUBROUTINE ARSME(M, M1, M2, ME, ND, RT, NI, NS, X, R, RL,          ARS   1Ø
     * IBV, LO, E, U, D)                                                 ARS   2Ø
C PURPOSE                                                               ARS   3Ø
C   TO SOLVE A MULTIPLE-RESOURCE NETWORK SCHEDULING PROBLEM -PREEMPTIVE ARS   4Ø
C   CASE BY THE REVISED SIMPLEX METHOD WITH THE PRODUCT FORM OF THE     ARS   5Ø
C   INVERSE. *ACTIVITIES-ON-ARCS* NETWORK REPRESENTATION IS USED.       ARS   6Ø
C   THIS IS THE PRIMARY SUBROUTINE AND IT COORDINATES THE SPECIAL       ARS   7Ø
C   PURPOSE SUBROUTINES GEN,KFIND,PRISET,ORDSNE.                        ARS   8Ø
C   THE SUBROUTINE USES A LINE-PRINTER AS PROGRAMMING UNIT 2 AND        ARS   9Ø
C   MAGNETIC TAPES AS PROGRAMMING UNIT 3.                               ARS  1ØØ
C   THE MAGNETIC TAPES ARE USED ONLY IF THE INVERSE EXCEEDS THE         ARS  11Ø
C   DIMENSION LIMIT OF VECTOR E.                                        ARS  12Ø
C DESCRIPTION OF PARAMETERS                                             ARS  13Ø
C   M    NUMBER OF ACTIVITIES                                           ARS  14Ø
C   M1   DIMENSION LIMIT OF VECTORS IBV,U,D AND X, EQUAL TO M+1         ARS  15Ø
C   M2   DIMENSION LIMIT OF VECTOR NS, EQUAL TO M+M+1       .           ARS  16Ø
C   ME   DIMENSION LIMIT OF VECTOR E                                    ARS  17Ø
C   ND   NUMBER OF NODES                                                ARS  18Ø
C   RT   NUMBER OF RESOURCE TYPES                                       ARS  19Ø
C   NI   DECLARED LIMIT OF THE NUMBER OF ITERATIONS ( ABOUT 2*M )       ARS  2ØØ
C   NS   NETWORK STRUCTURE VECTOR CONTAINING THE CONSECUTIVE ACTIVITIES ARS  21Ø
C        AS ORDERED PAIRS OF NODES.                                     ARS  22Ø
C        NO INITIAL VALUE IS REQUIRED FOR ELEMENT NS(M2).               ARS  23Ø
C   X    VECTOR OF ACTIVITY DURATIONS                                   ARS  24Ø
C   R    MATRIX OF RESOURCE REQUIREMENTS                                ARS  25Ø
C   RL   VECTOR OF RESOURCE LIMITS                                      ARS  26Ø
C   E    VECTOR OF THE INVERSE                                          ARS  27Ø
C   U,D  VECTORS USED FOR FINDING VECTORS INTRODUCED INTO AND           ARS  28Ø
C        ELIMINATED FROM THE BASIS RESPECTIVELY                         ARS  29Ø
C   LO   VECTOR OF THE INDICES OF VECTORS ELIMINATED FROM THE BASIS IN  ARS  3ØØ
C      SUCCESSIVE ITERATIONS                                            ARS  31Ø
C   IBV  VECTOR OF INDICES OF THE BASIC VECTORS                         ARS  32Ø
C   AS INPUT DATA PROVIDE M,M1,M2,ME,ND,RT,NI,NS,X,R,RL.                ARS  33Ø
```

```
      INTEGER RT, RT1                                                    ARS  34Ø
      LOGICAL PRE, PROC                                                  ARS  35Ø
      DIMENSION NS(M2), IBV(M1), LO(NI), E(ME), U(M1), D(M1),            ARS  36Ø
     * X(M1), R(M,RT), RL(RT)                                           ARS  37Ø
C FOR SOME COMPILERS OF FORTRAN THE DIMENSION LIMIT OF A MATRIX MUST     ARS  38Ø
C BE GREATER THAN 1, THEN IF RT IS TO BE EQUAL TO 1, MAKE RT=2 AND       ARS  39Ø
C RL(2)=Ø                                                                ARS  4ØØ
      PROC = .FALSE.                                                     ARS  41Ø
      PRE = .TRUE.                                                       ARS  42Ø
      RT1 = RT                                                           ARS  43Ø
      IF (RL(RT).EQ.Ø.) RT1 = RT - 1                                     ARS  44Ø
      LG = Ø                                                             ARS  45Ø
      IT = Ø                                                             ARS  46Ø
      JK = Ø                                                             ARS  47Ø
      IH2 = Ø                                                            ARS  48Ø
      IH = 1                                                             ARS  49Ø
      JP = 1                                                             ARS  5ØØ
      ND = ND - 1                                                        ARS  51Ø
      NS(M2) = NS(M2-1) + 1                                              ARS  52Ø
      IH1 = ME/M1                                                        ARS  53Ø
      RR = Ø.                                                            ARS  54Ø
      DO 1Ø I=1,M                                                        ARS  55Ø
        RR = RR + X(I)                                                   ARS  56Ø
   1Ø CONTINUE                                                           ARS  57Ø
      X(M1) = -RR                                                        ARS  58Ø
C CHECK IF THE RESOURCE REQUIREMENTS FOR ANY TYPE OF RESOURCE ARE        ARS  59Ø
C EQUAL TO EACH OTHER. IF SO, CALCULATE IP IT IS THE MAXIMUM NUMBER      ARS  6ØØ
C OF ACTIVITIES WHICH MAY BE PERFORMED SIMULTANEOUSLY.                   ARS  61Ø
      IP = 99999                                                         ARS  62Ø
      DO 3Ø J=1,RT1                                                      ARS  63Ø
        P = R(1,J)                                                       ARS  64Ø
        DO 2Ø I=2,M                                                      ARS  65Ø
          IF (P.NE.R(I,J)) GO TO 4Ø                                      ARS  66Ø
   2Ø   CONTINUE                                                         ARS  67Ø
        IF (P.LT.Ø.ØØØØØ1) GO TO 3Ø                                      ARS  68Ø
        I = (RL(J)+.ØØØØ1)/P                                             ARS  69Ø
        IF (IP.GT.I) IP = I                                              ARS  7ØØ
   3Ø CONTINUE                                                           ARS  71Ø
      IF (IP.NE.99999) PROC = .TRUE.                                     ARS  72Ø
   4Ø D(M1) = Ø.                                                         ARS  73Ø
C CALCULATE VECTOR U                                                     ARS  74Ø
      DO 5Ø I=1,M                                                        ARS  75Ø
        U(I) = Ø.                                                        ARS  76Ø
   5Ø CONTINUE                                                           ARS  77Ø
      U(M1) = 1.                                                         ARS  78Ø
      MF = IH*M1                                                         ARS  79Ø
      IF (LG.EQ.Ø) GO TO 8Ø                                              ARS  8ØØ
      LG = -1                                                            ARS  81Ø
      JK = IT                                                            ARS  82Ø
      JP = IH2 + 1                                                       ARS  83Ø
      GO TO 8Ø                                                           ARS  84Ø
   6Ø LG = 1                                                             ARS  85Ø
      JK = IH2                                                           ARS  86Ø
   7Ø IF (JK.LT.IH1) GO TO 13Ø                                           ARS  87Ø
      BACKSPACE 3                                                        ARS  88Ø
      BACKSPACE 3                                                        ARS  89Ø
      READ (3) E                                                         ARS  9ØØ
      MF = IH1*M1                                                        ARS  91Ø
      JP = JP - IH1                                                      ARS  92Ø
   8Ø J = JK                                                             ARS  93Ø
   9Ø IF (J.LT.JP) GO TO 11Ø                                             ARS  94Ø
      MF = MF - M1                                                       ARS  95Ø
      TH = Ø.                                                            ARS  96Ø
      DO 1ØØ I=1,M1                                                      ARS  97Ø
        MQ = MF + I                                                      ARS  98Ø
        TH = TH + U(I)*E(MQ)                                             ARS  99Ø
  1ØØ CONTINUE                                                           ARS 1ØØØ
      MQ = LO(J)                                                         ARS 1Ø1Ø
      U(MQ) = TH                                                         ARS 1Ø2Ø
      J = J - 1                                                          ARS 1Ø3Ø
      GO TO 9Ø                                                           ARS 1Ø4Ø
  11Ø IF (LG) 6Ø, 13Ø, 12Ø                                               ARS 1Ø5Ø
  12Ø JK = JK - IH1                                                      ARS 1Ø6Ø
      GO TO 7Ø                                                           ARS 1Ø7Ø
C FIND K, I.E. THE INDEX OF THE NEW OPTIMAL BASIC VECTOR                 ARS 1Ø8Ø
```

```
  13Ø CALL GEN(PROC, PRE, .FALSE., M, M1, M2, ND, K, J, RT, RT1,      ARS 1Ø9Ø
     * IP, NS, IBV, LO, U, D, R, RL)                                  ARS 11ØØ
C CHECK THE SOLUTION OPTIMALITY CRITERION                            ARS 111Ø
      IF (D(M1).LE.(-Ø.ØØØØ1)) GO TO 23Ø                             ARS 112Ø
C PRINT THE RESULTS OF ARSME                                         ARS 113Ø
      X(M1) = -X(M1)                                                 ARS 114Ø
      WRITE (2,99999) X(M1)                                          ARS 115Ø
      CALL ORDSNE(.TRUE., M, IBV, X)                                 ARS 116Ø
      DO 14Ø I=2,M                                                   ARS 117Ø
        X(I) = X(I) + X(I-1)                                         ARS 118Ø
  14Ø CONTINUE                                                       ARS 119Ø
      J = 1                                                          ARS 12ØØ
      K = 44                                                         ARS 121Ø
  15Ø IF (M.LT.K) K = M                                              ARS 122Ø
      I = 1                                                          ARS 123Ø
      JK = 1                                                         ARS 124Ø
      IF (J.GT.1) JK = K - 43                                        ARS 125Ø
      DO 16Ø I=JK,K                                                  ARS 126Ø
        LO(I) = I                                                    ARS 127Ø
  16Ø CONTINUE                                                       ARS 128Ø
      WRITE (2,99986)                                                ARS 129Ø
      JK = J                                                         ARS 13ØØ
      IF (J.GT.4) JK = 4                                             ARS 131Ø
      GO TO (18Ø, 19Ø, 2ØØ, 21Ø), JK                                 ARS 132Ø
  17Ø WRITE (2,99987)                                                ARS 133Ø
      IF (K.EQ.M) RETURN                                             ARS 134Ø
      J = J + 1                                                      ARS 135Ø
      K = K + 44                                                     ARS 136Ø
      GO TO 15Ø                                                      ARS 137Ø
  18Ø IF (M.GE.1Ø) WRITE (2,99997) (LO(I),I=1Ø,K,2)                  ARS 138Ø
      JK = 8                                                         ARS 139Ø
      IF (K.LT.8) JK = K                                             ARS 14ØØ
      WRITE (2,99996) (LO(I),I=1,K,2)                                ARS 141Ø
      WRITE (2,99995) (LO(I),I=2,JK,2)                               ARS 142Ø
      WRITE (2,99988)                                                ARS 143Ø
      GO TO 22Ø                                                      ARS 144Ø
  19Ø IF (M.GE.46) WRITE (2,99994) (LO(I),I=46,K,2)                  ARS 145Ø
      WRITE (2,99993) (LO(I),I=45,K,2)                               ARS 146Ø
      WRITE (2,99988)                                                ARS 147Ø
      GO TO 22Ø                                                      ARS 148Ø
  2ØØ IF (M.GE.9Ø) WRITE (2,99992) (LO(I),I=9Ø,K,2)                  ARS 149Ø
      WRITE (2,99991) (LO(I),I=89,K,2)                               ARS 15ØØ
      WRITE (2,99988)                                                ARS 151Ø
      GO TO 22Ø                                                      ARS 152Ø
  21Ø IF (M.GE.134) WRITE (2,9999Ø) (LO(I),I=134,K,2)                ARS 153Ø
      WRITE (2,99989) (LO(I),I=133,K,2)                              ARS 154Ø
      WRITE (2,99988)                                                ARS 155Ø
  22Ø CALL GEN(PROC, .TRUE., .TRUE., M, M1, M2, ND, K, J, RT, RT1,   ARS 156Ø
     * IP, NS, IBV, LO, U, X, R, RL)                                 ARS 157Ø
      GO TO 17Ø                                                      ARS 158Ø
  23Ø TH = 1Ø.E7Ø                                                    ARS 159Ø
      IF (.NOT.PRE) GO TO 25Ø                                        ARS 16ØØ
C FIND LO(1), I.E. THE INDEX OF THE VECTOR ELIMINATED FROM THE BASIS ARS 161Ø
C IN THE FIRST ITERATION                                             ARS 162Ø
      PRE = .FALSE.                                                  ARS 163Ø
      IT = 1                                                         ARS 164Ø
      JK = 1                                                         ARS 165Ø
      DO 24Ø I=1,M                                                   ARS 166Ø
        IF (D(I).EQ.Ø.Ø .OR. TH.LE.X(I)) GO TO 24Ø                   ARS 167Ø
        TH = X(I)                                                    ARS 168Ø
        L = I                                                        ARS 169Ø
  24Ø CONTINUE                                                       ARS 17ØØ
      LO(1) = L                                                      ARS 171Ø
      GO TO 38Ø                                                      ARS 172Ø
  25Ø IF (LG.EQ.Ø) GO TO 28Ø                                         ARS 173Ø
      LG = -1                                                        ARS 174Ø
      REWIND 3                                                       ARS 175Ø
      JP = 1 - IH1                                                   ARS 176Ø
      JK = IH1                                                       ARS 177Ø
  26Ø IF (JK.GT.IH2) GO TO 33Ø                                       ARS 178Ø
  27Ø JP = JP + IH1                                                  ARS 179Ø
      READ (3) E                                                     ARS 18ØØ
  28Ø MF = -M1                                                       ARS 181Ø
      J = JP                                                         ARS 182Ø
  29Ø IF (J.GT.JK) GO TO 31Ø                                         ARS 183Ø
      MF = MF + M1                                                   ARS 184Ø
```

```
C CALCULATE VECTOR D                                                    ARS 185Ø
      L = LO(J)                                                         ARS 186Ø
      RR = D(L)                                                         ARS 187Ø
      DO 3ØØ I=1,M                                                      ARS 188Ø
        MQ = MF + I                                                     ARS 189Ø
        D(I) = D(I) + RR*E(MQ)                                          ARS 19ØØ
  3ØØ CONTINUE                                                          ARS 191Ø
      D(L) = D(L) - RR                                                  ARS 192Ø
      J = J + 1                                                         ARS 193Ø
      GO TO 29Ø                                                         ARS 194Ø
  31Ø IF (LG) 32Ø, 34Ø, 34Ø                                             ARS 195Ø
  32Ø JK = JK + IH1                                                     ARS 196Ø
      GO TO 26Ø                                                         ARS 197Ø
  33Ø JK = IH2 + IH                                                     ARS 198Ø
      LG = 1                                                            ARS 199Ø
      GO TO 27Ø                                                         ARS 2ØØØ
C INCREASE THE NUMBER OF ITERATIONS IN THE ITERATION COUNTER            ARS 2Ø1Ø
  34Ø IT = IT + 1                                                       ARS 2Ø2Ø
      IF (IT.GT.NI) WRITE (2,99998)                                     ARS 2Ø3Ø
      IF (IT.GT.NI) STOP                                                ARS 2Ø4Ø
      IH = IH + 1                                                       ARS 2Ø5Ø
      IF (IH.LE.IH1) GO TO 36Ø                                          ARS 2Ø6Ø
      IF (LG.NE.Ø) GO TO 35Ø                                            ARS 2Ø7Ø
      REWIND 3                                                          ARS 2Ø8Ø
      WRITE (3) E                                                       ARS 2Ø9Ø
      LG = -1                                                           ARS 21ØØ
  35Ø IH = 1                                                            ARS 211Ø
      IH2 = IH2 + IH1                                                   ARS 212Ø
  36Ø IF (LG.EQ.Ø) JK = IH                                              ARS 213Ø
C FIND LO(IT), I.E. THE INDEX OF THE VECTOR ELIMINATED FROM THE         ARS 214Ø
C BASIS I ITERATION IT.                                                 ARS 215Ø
      DO 37Ø I=1,M                                                      ARS 216Ø
        IF (D(I).LE.Ø.) GO TO 37Ø                                       ARS 217Ø
        RR = X(I)/D(I)                                                  ARS 218Ø
        IF (TH.LE.RR) GO TO 37Ø                                         ARS 219Ø
        TH = RR                                                         ARS 22ØØ
        L = I                                                           ARS 221Ø
  37Ø CONTINUE                                                          ARS 222Ø
      LO(IT) = L                                                        ARS 223Ø
C CALCULATE VECTORS E AND X                                             ARS 224Ø
  38Ø MF = IH*M1 - M1                                                   ARS 225Ø
      MQ = MF + L                                                       ARS 226Ø
      RR = 1./D(L)                                                      ARS 227Ø
      E(MQ) = RR                                                        ARS 228Ø
      TH = X(L)                                                         ARS 229Ø
      X(L) = TH*RR                                                      ARS 23ØØ
      DO 39Ø I=1,M1                                                     ARS 231Ø
        IF (I.EQ.L) GO TO 39Ø                                           ARS 232Ø
        MQ = MF + I                                                     ARS 233Ø
        G = -D(I)*RR                                                    ARS 234Ø
        E(MQ) = G                                                       ARS 235Ø
        X(I) = X(I) + G*TH                                              ARS 236Ø
  39Ø CONTINUE                                                          ARS 237Ø
      IF (LG.EQ.Ø) GO TO 4ØØ                                            ARS 238Ø
      IF (IH.GT.1) BACKSPACE 3                                          ARS 239Ø
      WRITE (3) E                                                       ARS 24ØØ
C MEMORIZE THE INDEX OF THE VECTOR INTRODUCED INTO THE BASIS            ARS 241Ø
  4ØØ IBV(L) = K                                                        ARS 242Ø
      GO TO 4Ø                                                          ARS 243Ø
99999 FORMAT (//////35H   AUTOMATIC REVISED SIMPLEX METHOD///           ARS 244Ø
     * 17H OPTIMAL SOLUTION//33H MINIMAL SCHEDULE LENGTH   TMIN =,       ARS 245Ø
     * F12.5///)                                                        ARS 246Ø
99998 FORMAT (/////37H NUMBER OF ITERATIONS GREATER THAN NI///)         ARS 247Ø
99997 FORMAT (3ØX, 1H*, 17X, 18I4)                                      ARS 248Ø
99996 FORMAT (3ØX, 2H* , I2, 21I4)                                      ARS 249Ø
99995 FORMAT (1H+, 31X, 4I4)                                            ARS 25ØØ
99994 FORMAT (3ØX, 2H* , 22I4)                                          ARS 251Ø
99993 FORMAT (3ØX, 2H* , I2, 21I4)                                      ARS 252Ø
99992 FORMAT (3ØX, 2H* , 5I4, 1X, 17I4)                                 ARS 253Ø
99991 FORMAT (3ØX, 2H* , I2, 5I4, 1X, 16I4)                             ARS 254Ø
9999Ø FORMAT (3ØX, 3H*  , 22I4)                                         ARS 255Ø
99989 FORMAT (3ØX, 1H*, 22I4)                                           ARS 256Ø
99988 FORMAT (3ØX, 1H*/1X, 12Ø(1H-)/3ØX, 1H*)                           ARS 257Ø
99987 FORMAT (3ØX, 1H*/1X, 12Ø(1H-)//)                                  ARS 258Ø
99986 FORMAT (1X, 12Ø(1H-)/3ØX, 1H*/3ØX, 1H*, 4ØX, 1ØHACTIVITIES/       ARS 259Ø
```

```
     * 3ØX, 1H*/13X, 4HTIME, 13X, 2H* , 89(1H-)/11X, 9HINTERVALS,      ARS 26ØØ
     * 1ØX, 1H*)                                                       ARS 261Ø
      END                                                              ARS 262Ø


      SUBROUTINE GEN(PROC, PRE, PRI, M, M1, M2, ND, K, KW, RT, RT1,    GEN   1Ø
     * IP, NS, IBV, LO, U, D, R, RL)                                   GEN   2Ø
C PURPOSE                                                              GEN   3Ø
C   SUBROUTINE GEN GENERATES ALL RESOURCE FEASIBLE SETS.               GEN   4Ø
C DESCRIPTION OF PARAMETERS                                            GEN   5Ø
C   PROC LOGICAL VARIABLE. IF PROC IS .TRUE. RESOURCE CONSTRAINTS ARE  GEN   6Ø
C        NOT CHECKED, AND SETS WITH THE NUMBER OF ELEMENTS LESS THAN OR GEN   7Ø
C        EQUAL TO IP ARE GENERATED. IP IS DETERMINED IN SUBROUT. ARSME GEN   8Ø
C   PRE  LOGICAL VARIABLE. PRE IS .TRUE. IN THE FIRST ITERATION        GEN   9Ø
C   PRI  LOGICAL VARIABLE. IF PRI IS .TRUE. SUBROUTINE GEN CALLS       GEN  1ØØ
C        SUBROUTINE PRISET TO PRINT RESULTS OF ARSME                   GEN  11Ø
C   DA   LOGICAL VARIABLE. DA IS .TRUE. IF THE MAIN SET IS RESOURCE    GEN  12Ø
C        FEASIBLE                                                      GEN  13Ø
C   KAS  VECTOR CONTAINING THE NUMBERS OF ACTIVITIES IN THE MAIN SET   GEN  14Ø
C        CURRENTLY BEING CONSIDERED.                                   GEN  15Ø
C   KAR  VECTOR OF THOSE INDICES OF KAS ELEMENTS WHICH COMPOSE THE     GEN  16Ø
C        SUBSET CURRENTLY BEING CONSIDERED.                            GEN  17Ø
C   IC   NUMBER OF ELEMENTS OF THE MAIN SET CURRENTLY BEING CONSIDERED GEN  18Ø
C   NR   NUMBER OF ELEMENTS OF THE SUBSET CURRENTLY BEING CONSIDERED   GEN  19Ø
C   LI   SET COUNTER                                                  GEN  2ØØ
      INTEGER RT, RT1                                                  GEN  21Ø
      LOGICAL PRE, PRI, DA, PROC                                       GEN  22Ø
      DIMENSION NS(M2), IBV(M1), LO(M), U(M1), D(M1), R(M,RT),         GEN  23Ø
     * RL(RT), KAS(1ØØ), KAR(1ØØ), KA(1ØØ), RC(1ØØ)                   GEN  24Ø
      JS = 1                                                           GEN  25Ø
      LI = 1                                                           GEN  26Ø
      DO 3ØØ KG=1,ND                                                   GEN  27Ø
C GENERATE A MAIN SET                                                  GEN  28Ø
        IC = Ø                                                         GEN  29Ø
        DO 1Ø ID=1,M2,2                                                GEN  3ØØ
          IF (NS(ID).GE.KG) GO TO 2Ø                                   GEN  31Ø
          IF (NS(ID+1).LE.KG) GO TO 1Ø                                 GEN  32Ø
          IC = IC + 1                                                  GEN  33Ø
          KAS(IC) = (ID+1)/2                                           GEN  34Ø
          KAR(IC) = IC                                                 GEN  35Ø
   1Ø   CONTINUE                                                       GEN  36Ø
C NOTE-- DO LOOP IS NEVER EXITED HERE.                                 GEN  37Ø
   2Ø   II = IC                                                        GEN  38Ø
        J = ID                                                         GEN  39Ø
        DO 3Ø ID=J,M2,2                                                GEN  4ØØ
          IF (NS(ID).NE.KG) GO TO 4Ø                                   GEN  41Ø
          IC = IC + 1                                                  GEN  42Ø
          KAS(IC) = (ID+1)/2                                           GEN  43Ø
          KAR(IC) = IC                                                 GEN  44Ø
   3Ø   CONTINUE                                                       GEN  45Ø
   4Ø   IF (PROC) GO TO 12Ø                                            GEN  46Ø
C CHECK THE RESOURCE FEASIBILITY OF THE MAIN SET                       GEN  47Ø
        NP = 99999                                                     GEN  48Ø
        DA = .TRUE.                                                    GEN  49Ø
        DO 9Ø J=1,RT1                                                  GEN  5ØØ
          G = Ø.                                                       GEN  51Ø
          DO 5Ø I=1,IC                                                 GEN  52Ø
            MQ = KAS(I)                                                GEN  53Ø
            H = R(MQ,J)                                                GEN  54Ø
            G = G + H                                                  GEN  55Ø
            RC(I) = H                                                  GEN  56Ø
   5Ø     CONTINUE                                                     GEN  57Ø
          IF (G.LE.RL(J)) GO TO 9Ø                                     GEN  58Ø
          DA = .FALSE.                                                 GEN  59Ø
C ORDER ACTIVITIES IN THE MAIN SET                                     GEN  6ØØ
          IF (II.GT.1) CALL ORDSNE(.FALSE., II, KAS, RC)               GEN  61Ø
          IF ((IC-II).GT.1) CALL ORDSNE(.FALSE., IC-II, KAS(II+1),     GEN  62Ø
     *     RC(II+1))                                                   GEN  63Ø
          G = Ø.                                                       GEN  64Ø
          DO 6Ø I=1,IC                                                 GEN  65Ø
            G = G + RC(I)                                              GEN  66Ø
            IF (G.GT.RL(J)) GO TO 7Ø                                   GEN  67Ø
   6Ø     CONTINUE                                                     GEN  68Ø
C NOTE-- DO LOOP IS NEVER EXITED HERE.                                 GEN  69Ø
   7Ø     IF (NP.LE.I) GO TO 9Ø                                        GEN  7ØØ
          NP = I                                                       GEN  71Ø
```

```
            IF (RT1.EQ.1) GO TO 9Ø                                    GEN  72Ø
            DO 8Ø I=1,IC                                              GEN  73Ø
              KA(I) = KAS(I)                                          GEN  74Ø
     8Ø     CONTINUE                                                  GEN  75Ø
     9Ø   CONTINUE                                                    GEN  76Ø
          IF (DA) GO TO 12Ø                                           GEN  77Ø
          IF (RT1.EQ.1) GO TO 11Ø                                     GEN  78Ø
          DO 1ØØ I=1,IC                                               GEN  79Ø
            KAS(I) = KA(I)                                            GEN  8ØØ
    1ØØ   CONTINUE                                                    GEN  81Ø
  C GENERATE SUBSETS                                                  GEN  82Ø
    11Ø   NR = IC                                                     GEN  83Ø
          IF (NP.LE.(II+1)) GO TO 25Ø                                 GEN  84Ø
    12Ø   NR = II                                                     GEN  85Ø
    13Ø   NR = NR + 1                                                 GEN  86Ø
    14Ø   IF (PROC) GO TO 27Ø                                         GEN  87Ø
          IF (DA) GO TO 18Ø                                           GEN  88Ø
  C CHECK RESOURCE FEASIBILITY OF THE SUBSET                          GEN  89Ø
          NP = 99999                                                  GEN  9ØØ
          DO 17Ø J=1,RT1                                              GEN  91Ø
            G = Ø.                                                    GEN  92Ø
            DO 15Ø I=1,NR                                             GEN  93Ø
              MQ = KAR(I)                                             GEN  94Ø
              MQ = KAS(MQ)                                            GEN  95Ø
              G = G + R(MQ,J)                                         GEN  96Ø
              IF (G.GT.RL(J)) GO TO 16Ø                               GEN  97Ø
    15Ø     CONTINUE                                                  GEN  98Ø
            GO TO 17Ø                                                 GEN  99Ø
    16Ø     IF (NP.GT.I) NP = I                                       GEN 1ØØØ
    17Ø   CONTINUE                                                    GEN 1Ø1Ø
          IF (NR.GE.NP) GO TO 25Ø                                     GEN 1Ø2Ø
  C GO TO SIMPLEX PROCEDURE                                           GEN 1Ø3Ø
    18Ø   IF (PRI) GO TO 26Ø                                          GEN 1Ø4Ø
          CALL KFIND(PRE, M1, NR, K, LI, KAS, KAR, IBV, U, D)         GEN 1Ø5Ø
  C INCREASE THE NUMBER OF SETS IN THE SET COUNTER                    GEN 1Ø6Ø
    19Ø   LI = LI + 1                                                 GEN 1Ø7Ø
          IF (KAR(NR).LT.IC) GO TO 13Ø                                GEN 1Ø8Ø
  C GENERATE THE PRIMARY SUBSET                                       GEN 1Ø9Ø
    2ØØ   NR = NR - 1                                                 GEN 11ØØ
          IF (NR.EQ.Ø) GO TO 3ØØ                                      GEN 111Ø
    21Ø   KAR(NR) = KAR(NR) + 1                                       GEN 112Ø
          IF (KAR(NR).EQ.IC) GO TO 14Ø                                GEN 113Ø
          MQ = NR + 1                                                 GEN 114Ø
          MP = NR + IC - KAR(NR)                                      GEN 115Ø
          IF (PROC) GO TO 29Ø                                         GEN 116Ø
    22Ø   DO 23Ø I=MQ,MP                                              GEN 117Ø
            KAR(I) = KAR(I-1) + 1                                     GEN 118Ø
    23Ø   CONTINUE                                                    GEN 119Ø
    24Ø   IF (KAR(NR).LE.II) NR = II - KAR(NR) + NR + 1                GEN 12ØØ
          GO TO 14Ø                                                   GEN 121Ø
  C GENERATE THE NEW PRIMARY SUBSET BECAUSE OF RESOURCE INFEASIBILITY GEN 122Ø
  C OF THE LAST-GENERATED SET                                         GEN 123Ø
    25Ø   IF (KAR(NP).EQ.IC) GO TO 2ØØ                               GEN 124Ø
          NR = NP                                                     GEN 125Ø
          GO TO 21Ø                                                   GEN 126Ø
  C CHECK IF THE SET COMPRISE THE OPTIMAL SOLUTION                    GEN 127Ø
    26Ø   IF (LI.NE.IBV(JS)) GO TO 19Ø                                GEN 128Ø
          CALL PRISET(M, NR, KW, JS, K, KAR, KAS, LO, D)              GEN 129Ø
          JS = JS + 1                                                 GEN 13ØØ
          IF (JS.EQ.M1) RETURN                                        GEN 131Ø
          GO TO 19Ø                                                   GEN 132Ø
  C GENERATE THE SETS IN THE CASE OF EQUAL RESOURCE REQUIREMENTS      GEN 133Ø
    27Ø   IF (NR.LE.IP) GO TO 18Ø                                     GEN 134Ø
          NR = IP                                                     GEN 135Ø
          IF (KAR(NR).LE.II) GO TO 28Ø                                GEN 136Ø
          GO TO 21Ø                                                   GEN 137Ø
    28Ø   KAR(NR) = II + 1                                            GEN 138Ø
          GO TO 18Ø                                                   GEN 139Ø
    29Ø   IF (MP.GT.IP) MP = IP                                       GEN 14ØØ
          IF (MP.LT.MQ) GO TO 24Ø                                     GEN 141Ø
          GO TO 22Ø                                                   GEN 142Ø
    3ØØ CONTINUE                                                      GEN 143Ø
        RETURN                                                        GEN 144Ø
        END                                                           GEN 145Ø
```

```
      SUBROUTINE KFIND(PRE, M1, NR, K, LI, KAS, KAR, IBV, U, D)          KFI  1Ø
C PURPOSE                                                                KFI  2Ø
C   SUBROUTINE KFIND CHECKS TO DETERMINE IF THE INTRODUCTION OF THE      KFI  3Ø
C   CURRENTLY CONSIDERED SET ( VECTOR ) INTO THE BASIS IS MORE           KFI  4Ø
C   PROFITABLE THAN THE INTRODUCTION OF THE PREVIOUSLY GENERATED SETS.   KFI  5Ø
      LOGICAL PRE                                                        KFI  6Ø
      DIMENSION U(M1), D(M1), IBV(M1), KAS(1ØØ), KAR(1ØØ)                KFI  7Ø
      IF (.NOT.(PRE .AND. (NR.EQ.1))) GO TO 1Ø                           KFI  8Ø
C CALCULATE THE INITIAL VALUES OF VECTOR IBV                             KFI  9Ø
      MQ = KAR(1)                                                        KFI 1ØØ
      MQ = KAS(MQ)                                                       KFI 11Ø
      IBV(MQ) = LI                                                       KFI 12Ø
C CALCULATE VALUE G OF THE CRITERION FOR INTRODUCING A VECTOR INTO       KFI 13Ø
C THE BASIS                                                              KFI 14Ø
   1Ø G = Ø.                                                             KFI 15Ø
      DO 2Ø I=1,NR                                                       KFI 16Ø
        MQ = KAR(I)                                                      KFI 17Ø
        MQ = KAS(MQ)                                                     KFI 18Ø
        G = G + U(MQ)                                                    KFI 19Ø
   2Ø CONTINUE                                                           KFI 2ØØ
      RR = 1 - NR                                                        KFI 21Ø
      G = G + U(M1)*RR                                                   KFI 22Ø
      IF (D(M1).LE.G) RETURN                                             KFI 23Ø
C MEMORIZE THE VECTOR FOR WHICH G IS AT MINIMUM                          KFI 24Ø
      D(M1) = G                                                          KFI 25Ø
      K = LI                                                             KFI 26Ø
      MQ = M1 - 1                                                        KFI 27Ø
      DO 3Ø I=1,MQ                                                       KFI 28Ø
        D(I) = Ø.                                                        KFI 29Ø
   3Ø CONTINUE                                                           KFI 3ØØ
      DO 4Ø I=1,NR                                                       KFI 31Ø
        MQ = KAR(I)                                                      KFI 32Ø
        MQ = KAS(MQ)                                                     KFI 33Ø
        D(MQ) = 1.                                                       KFI 34Ø
   4Ø CONTINUE                                                           KFI 35Ø
      RETURN                                                             KFI 36Ø
      END                                                                KFI 37Ø


      SUBROUTINE PRISET(M, NR, J, JS, K, KAR, KAS, LO, X)                PRI  1Ø
C PURPOSE                                                                PRI  2Ø
C   SUBROUTINE PRISET PRINTS THE SET OF ACTIVITIES COMPOSING THE         PRI  3Ø
C   OPTIMAL SOLUTION AND THE TIME-INTERVALS OF ITS PERFORMANCE.          PRI  4Ø
      DIMENSION KAR(1ØØ), KAS(1ØØ), LO(M), X(M)                          PRI  5Ø
      DO 1Ø I=1,M                                                        PRI  6Ø
        LO(I) = Ø                                                        PRI  7Ø
   1Ø CONTINUE                                                           PRI  8Ø
      DO 2Ø I=1,NR                                                       PRI  9Ø
        MQ = KAR(I)                                                      PRI 1ØØ
        MQ = KAS(MQ)                                                     PRI 11Ø
        MP = J*44                                                        PRI 12Ø
        MR = MP - 44                                                     PRI 13Ø
        IF (MQ.LE.MR .OR. MQ.GT.MP) GO TO 2Ø                             PRI 14Ø
        LO(MQ) = 1                                                       PRI 15Ø
   2Ø CONTINUE                                                           PRI 16Ø
      C = Ø.                                                             PRI 17Ø
      IF (JS.EQ.1) WRITE (2,99999) C, X(JS), (LO(I),I=1,K)               PRI 18Ø
      MQ = JS - 1                                                        PRI 19Ø
      IF (JS.NE.1) WRITE (2,99999) X(MQ), X(JS), (LO(I),I=1,K)           PRI 2ØØ
      RETURN                                                             PRI 21Ø
99999 FORMAT (1X, F12.5, 3H  -, F12.5, 4H  * , 44I2)                     PRI 22Ø
      END                                                                PRI 23Ø


      SUBROUTINE ORDSNE(PRI, IC, KAS, RC)                                ORD  1Ø
C PURPOSE                                                                ORD  2Ø
C   IF PRI IS .TRUE. SUBROUTINE ORDSNE ORDERS ELEMENTS OF VECTOR KAS     ORD  3Ø
C   IN INCREASING ORDER OF THEIR VALUES AND VECTOR RC CORRESPONDINGLY.   ORD  4Ø
C   IF PRI IS .FALSE. SUBROUTINE ORDSNE ORDERS ELEMENTS OF VECTOR RC     ORD  5Ø
C   IN DECREASING ORDER OF THEIR VALUES AND VECTOR KAS CORRESPONDINGLY.  ORD  6Ø
      LOGICAL PRI                                                        ORD  7Ø
      DIMENSION RC(IC), KAS(IC)                                          ORD  8Ø
      ID = -IC/2                                                         ORD  9Ø
   1Ø IF (ID.GE.Ø) RETURN                                                ORD 1ØØ
      MQ = ID + IC                                                       ORD 11Ø
```

```
      DO 4Ø J=1,MQ                                    ORD  12Ø
        I = J                                         ORD  13Ø
2Ø    IF (I.LT.1) GO TO 4Ø                            ORD  14Ø
      JJ = I - ID                                     ORD  15Ø
      IF (PRI) GO TO 5Ø                               ORD  16Ø
      IF (RC(I).GE.RC(JJ)) GO TO 4Ø                   ORD  17Ø
3Ø    G = RC(I)                                       ORD  18Ø
      RC(I) = RC(JJ)                                  ORD  19Ø
      RC(JJ) = G                                      ORD  2ØØ
      II = KAS(I)                                     ORD  21Ø
      KAS(I) = KAS(JJ)                                ORD  22Ø
      KAS(JJ) = II                                    ORD  23Ø
      I = I + ID                                      ORD  24Ø
      GO TO 2Ø                                        ORD  25Ø
4Ø CONTINUE                                           ORD  26Ø
      ID = ID/2                                       ORD  27Ø
      GO TO 1Ø                                        ORD  28Ø
5Ø IF (KAS(I).LE.KAS(JJ)) GO TO 4Ø                    ORD  29Ø
      GO TO 3Ø                                        ORD  3ØØ
      END                                             ORD  31Ø
```

ALGORITHM 521
Repeated Integrals of the Coerror Function [S15]

WALTER GAUTSCHI
Purdue University

Key Words and Phrases: repeated integrals of the coerror function, Taylor's series, three-term recurrence relation, Miller's backward recurrence algorithm
CR Categories: 5.12
Language: Fortran

DESCRIPTION

This algorithm implements the procedures developed in [1].

REFERENCES

1. GAUTSCHI, W. Evaluation of the repeated integrals of the coerror function. *ACM Trans. Math. Software 3*, 3 (Sept. 1977), 240–252.

ALGORITHM

```
      SUBROUTINE INERFC(X, NMAX, ACC, FZERO, F, IFLAG)                  INE   1Ø
C THIS PROGRAM GENERATES FN(X) FOR N=1,2,...,NMAX, WHERE                INE   2Ø
C FN(X)=EXP(X**2)*INERFC(X), IF X.GE.Ø., AND FN(X)=INERFC(X), IF        INE   3Ø
C X.LT.Ø., INERFC(X) BEING THE N-TH REPEATED INTEGRAL OF THE COERROR    INE   4Ø
C FUNCTION ERFC(X), EXTENDED FROM X TO INFINITY. THE PROGRAM ALSO       INE   5Ø
C SUPPLIES EXP(X**2)*ERFC(X),IF X.GE.Ø., AND ERFC(X), IF X.LT.Ø..       INE   6Ø
C THE USER HAS TO SPECIFY THE DESIRED ACCURACY, AS WELL AS THE PRECISIONINE   7Ø
C OF THE HIGHEST PRECISION MODE HE IS PREPARED TO USE. THE FORMER IS    INE   8Ø
C INPUT VIA THE PARAMETER ACC IN THE CALL LIST, WHICH DESIGNATES THE    INE   9Ø
C NUMBER OF CORRECT SIGNIFICANT DECIMAL DIGITS DESIRED IN THE RESULTS.  INE  1ØØ
C THE LATTER IS INPUT VIA THE PARAMETER PREC IN THE DATA STATEMENT. THE INE  11Ø
C VALUE OF PREC SHOULD BE SET APPROXIMATELY EQUAL TO BETA*ALOG(2.)/     INE  12Ø
C ALOG(1Ø.), WHERE BETA IS THE NUMBER OF BINARY DIGITS AVAILABLE IN THE INE  13Ø
C MANTISSA OF THE HIGHEST PRECISION FLOATING-POINT WORD. ORDINARILY, THEINE  14Ø
C HIGHEST PRECISION IS DOUBLE PRECISION, EITHER HARDWARE GENERATED OR   INE  15Ø
C SOFTWARE GENERATED, BUT IT MAY ALSO BE SINGLE PRECISION. IN ANY CASE, INE  16Ø
C ACC HAS TO BE LESS THAN PREC, THE DIFFERENCE BEING AT LEAST EQUAL TO  INE  17Ø
C ONE, PREFERABLY SEVERAL UNITS LARGER. EVIDENTLY, ACC SHOULD NOT BE    INE  18Ø
C SPECIFIED TO EXCEED THE PRECISION OF THE LOWEST PRECISION MODE USED.  INE  19Ø
C   THE DATA STATEMENT CONTAINS ANOTHER MACHINE-DEPENDENT PARAMETER,    INE  2ØØ
C BOTEXP, WHICH IS, APPROXIMATELY, THE SMALLEST NEGATIVE NUMBER SUCH    INE  21Ø
C THAT 1Ø.**BOTEXP IS STILL REPRESENTABLE ON THE COMPUTER IN SINGLE PRE-INE  22Ø
C CISION FLOATING-POINT ARITHMETIC. THE TYPE OF BOTEXP IS REAL. IN THE  INE  23Ø
C DATA STATEMENT BELOW THE PARAMETERS PREC AND BOTEXP ARE SET TO CORRE- INE  24Ø
C SPOND TO THE MACHINE CHARACTERISTICS OF THE CDC 65ØØ COMPUTER.        INE  25Ø
C   PARAMETER LIST                                                      INE  26Ø
C       X - - THE ARCUMENT OF INERFC. TYPE REAL.                        INE  27Ø
C    NMAX - - THE UPPER LIMIT ON N. TYPE INTEGER.                       INE  28Ø
C     ACC - - THE NUMBER OF CORRECT SIGNIFICANT DECIMAL DIGITS DESIRED  INE  29Ø
C             IN THE RESULTS. TYPE REAL.                                INE  3ØØ
```

Received 24 February 1976.

Author's address: Department of Computer Sciences, Purdue University, Lafayette, IN 47907.

```
C   FZERO - - AN OUTPUT VARIABLE RETURNING EXP(X**2)*ERFC(X), IF        INE  31Ø
C            X.GE.Ø., AND ERFC(X), IF X.LT.Ø. TYPE REAL.               INE  32Ø
C       F - - THE NAME OF A ONE-DIMENSIONAL ARRAY HOLDING THE RESULTS   INE  33Ø
C            FN(X), N=1,2,...,NMAX. THE ARRAY HAS DIMENSION NMAX.       INE  34Ø
C   IFLAG - - AN ERROR FLAG INDICATING A NUMBER OF FATAL ERROR CON-     INE  35Ø
C            DITIONS UPON EXECUTION. TYPE INTEGER. THE VALUES OF THIS   INE  36Ø
C            VARIABLE HAVE THE FOLLOWING MEANINGS.                      INE  37Ø
C            IFLAG=Ø - NO ERROR CONDITION.                              INE  38Ø
C            IFLAG=1 - ILLEGAL NEGATIVE OR ZERO VALUE OF NMAX.          INE  39Ø
C            IFLAG=2 - EXCESSIVELY LARGE VALUE OF NMAX. REDIMENSION THEINE  4ØØ
C                      ARRAYS RØ AND R1 TO HAVE DIMENSION .GE. NMAX, IFINE  41Ø
C                      INDEED NMAX MUST BE THAT LARGE.                  INE  42Ø
C            IFLAG=3 - INAPPROPRIATE ACCURACY SPECIFICATION.            INE  43Ø
C            IFLAG=4 - UNUSUAL UNDERFLOW OF EXP(-X**2). TRY REDUCING    INE  44Ø
C                      THE ERROR TOLERANCE PREC-ACC, EITHER BY DE-      INE  45Ø
C                      CREASING PREC OR BY INCREASING ACC.              INE  46Ø
C            IFLAG=5 - TAYLOR SERIES FOR ERFC(X) DOES NOT CONVERGE      INE  47Ø
C                      WITHIN 2ØØ TERMS FOR SOME UNKNOWN REASON.        INE  48Ø
C            IFLAG=6 - NU IN THE BACKWARD RECURRENCE ALGORITHM EXCEEDS  INE  49Ø
C                      5ØØØ, PROBABLY BECAUSE X IS A SMALL POSITIVE     INE  5ØØ
C                      NUMBER, NMAX RELATIVELY LARGE, AND THE ERROR     INE  51Ø
C                      TOLERANCE SMALL. TRY INCREASING THE ERROR TOLER-INE  52Ø
C                      ANCE PREC-ACC, EITHER BY INCREASING PREC OR BY   INE  53Ø
C                      DECREASING ACC.                                  INE  54Ø
C    THE SUBROUTINE BELOW IS WRITTEN WITH THE ASSUMPTION THAT PART OF    INE  55Ø
C THE COMPUTATION IS DONE IN DOUBLE PRECISION AND THE REST IN SINGLE     INE  56Ø
C PRECISION. TO OBTAIN A VERSION OF THIS SUBROUTINE, THAT OPERATES       INE  57Ø
C ENTIRELY IN SINGLE PRECISION, THE FOLLOWING CHANGES MUST BE MADE.      INE  58Ø
C   (1) DELETE THE DOUBLE PRECISION TYPE DECLARATION.                    INE  59Ø
C   (2) REPLACE THE SECTION OF PROGRAM FROM STATEMENT LABELED 3Ø         INE  6ØØ
C (INCLUSIVE) TO STATEMENT LABELED 6Ø (INCLUSIVE) BY THE FOLLOWING       INE  61Ø
C PIECE OF PROGRAM.                                                      INE  62Ø
C          3Ø DEPS=.5*1Ø.**(-PREC)                                       INE  63Ø
C             TE=C*X                                                     INE  64Ø
C             SUM=1.-TE                                                  INE  65Ø
C             N=Ø                                                        INE  66Ø
C             N1=1                                                       INE  67Ø
C          4Ø N=N+1                                                      INE  68Ø
C             N1=N1+2                                                    INE  69Ø
C             IF(N.GT.2ØØ) GOTO 27Ø                                      INE  7ØØ
C             TE=-TE*XSQ/FLOAT(N)                                        INE  71Ø
C             TERM=TE/FLOAT(N1)                                          INE  72Ø
C             SUM=SUM-TERM                                               INE  73Ø
C             IF(ABS(TERM).GT.DEPS*ABS(SUM)) GOTO 4Ø                     INE  74Ø
C             F1=T*SUM                                                   INE  75Ø
C    THE ARRAYS RØ AND R1 IN THE DIMENSION STATEMENT ARE LOCAL TO THE    INE  76Ø
C SUBROUTINE. THEY BOTH NEED TO HAVE DIMENSION LARGER OR EQUAL TO NMAX.  INE  77Ø
C WE DECLARED THEIR DIMENSION TO BE 5ØØ, ASSUMING THAT NMAX WILL NOT     INE  78Ø
C EXCEED 5ØØ. IN THE EVENT IT DOES, THE SUBROUTINE EXITS WITH THE ERROR  INE  79Ø
C FLAG AT 2. IF THE USER EXPECTS HIS VALUES OF NMAX TO BE CONSISTENTLY   INE  8ØØ
C MUCH LESS THAN 5ØØ, HE CAN SAVE STORAGE BY LOWERING THE DIMENSION OF   INE  81Ø
C RØ AND R1 AND AT THE SAME TIME ADJUSTING THE ERROR EXIT STATEMENT (THEINE  82Ø
C THIRD EXECUTABLE STATEMENT IN THE PROGRAM).                            INE  83Ø
C    WITH THE EXCEPTION NOTED UNDER IFLAG=4, ANY VARIABLE THAT UNDER-    INE  84Ø
C FLOWS MAY BE SET EQUAL TO ZERO. NO PRECAUTION IS TAKEN AGAINST OVER-   INE  85Ø
C FLOW, WHICH MAY OCCUR IN THE DO-LOOP AFTER STATEMENT 7Ø IF X.LT.Ø.     INE  86Ø
C AND ABS(X) AND/OR NMAX IS LARGE.                                       INE  87Ø
C    REFERENCE - W. GAUTSCHI, EVALUATION OF THE REPEATED INTEGRALS OF    INE  88Ø
C THE COERROR FUNCTION, ACM TRANS. MATH. SOFTWARE.                       INE  89Ø
      DIMENSION F(NMAX), RØ(5ØØ), R1(5ØØ)                               INE  9ØØ
      DOUBLE PRECISION DEPS, DC, DX, DXSQ, DFM1, DFØ, DF1, DN, DN1,     INE  91Ø
     *  DTE, DTERM, DSUM                                                INE  92Ø
      LOGICAL FRSTTM                                                     INE  93Ø
      DATA FRSTTM, PREC, BOTEXP /.TRUE.,28.8989,-293./                   INE  94Ø
      IFLAG = Ø                                                          INE  95Ø
      IF (NMAX.LT.1) GO TO 23Ø                                           INE  96Ø
      IF (NMAX.GT.5ØØ) GO TO 24Ø                                         INE  97Ø
      TOL = PREC - ACC                                                   INE  98Ø
      IF (TOL.LT.1.) GO TO 25Ø                                           INE  99Ø
      I = 1                                                              INE 1ØØØ
      EPS = .5*1Ø.**(-ACC)                                               INE 1Ø1Ø
      XSQ = X*X                                                          INE 1Ø2Ø
      IF (.NOT.FRSTTM) GO TO 1Ø                                          INE 1Ø3Ø
      P = ATAN(1.)                                                       INE 1Ø4Ø
      AL1Ø = ALOG(1Ø.)                                                   INE 1Ø5Ø
      C = SQRT(1./P)                                                     INE 1Ø6Ø
      DC = DSQRT(1.DØ/DATAN(1.DØ))                                       INE 1Ø7Ø
```

```
      FRSTTM = .FALSE.                                                  INE 1Ø8Ø
   1Ø S = Ø.                                                            INE 1Ø9Ø
      IF (-XSQ.GT.AL1Ø*BOTEXP) S = EXP(-XSQ)                            INE 11ØØ
      B = .5*AL1Ø*TOL + .25*ALOG(2.*P)                                  INE 111Ø
      B1 = .5*AL1Ø*(TOL-1.)                                             INE 112Ø
      IF (XSQ.GT.B) GO TO 9Ø                                            INE 113Ø
      IF (X.LT.Ø.) GO TO 2Ø                                             INE 114Ø
      FØ = C                                                            INE 115Ø
      IF (S.EQ.Ø.) GO TO 26Ø                                            INE 116Ø
      T = 1./S                                                          INE 117Ø
      SQ = SQRT(2.*FLOAT(NMAX))                                         INE 118Ø
      IF ((X.GE.1.12 .AND. XSQ+SQ*X.GT.B) .OR. (X.LT.1.12 .AND.         INE 119Ø
     *  SQ*X.GT.B1)) GO TO 1ØØ                                          INE 12ØØ
      GO TO 3Ø                                                          INE 121Ø
   2Ø FØ = C*S                                                          INE 122Ø
      T = 1.                                                            INE 123Ø
C                                                                       INE 124Ø
C EVALUATION OF ERFC(X) BY TAYLOR SERIES IN DOUBLE PRECISION            INE 125Ø
C                                                                       INE 126Ø
   3Ø DEPS = .5DØ*1Ø.DØ**(-PREC)                                        INE 127Ø
      DX = DBLE(X)                                                      INE 128Ø
      DXSQ = DX*DX                                                      INE 129Ø
      DTE = DC*DX                                                       INE 13ØØ
      DSUM = 1.DØ - DTE                                                 INE 131Ø
      DN = Ø.DØ                                                         INE 132Ø
      DN1 = 1.DØ                                                        INE 133Ø
   4Ø DN = DN + 1.DØ                                                    INE 134Ø
      DN1 = DN1 + 2.DØ                                                  INE 135Ø
      IF (DN.GT.2.D2) GO TO 27Ø                                         INE 136Ø
      DTE = -DTE*DXSQ/DN                                                INE 137Ø
      DTERM = DTE/DN1                                                   INE 138Ø
      DSUM = DSUM - DTERM                                               INE 139Ø
      IF (DABS(DTERM).GT.DEPS*DABS(DSUM)) GO TO 4Ø                      INE 14ØØ
      IF (X.LT.Ø.) GO TO 6Ø                                             INE 141Ø
C                                                                       INE 142Ø
C EVALUATION OF EXP(X**2)*INERFC(X),X.GE.Ø.,BY FORWARD RECURSION IN     INE 143Ø
C DOUBLE PRECISION                                                      INE 144Ø
C                                                                       INE 145Ø
      DFØ = DC                                                          INE 146Ø
      DF1 = DEXP(DXSQ)*DSUM                                             INE 147Ø
      FZERO = SNGL(DF1)                                                 INE 148Ø
      DO 5Ø N=1,NMAX                                                    INE 149Ø
        DFM1 = DFØ                                                      INE 15ØØ
        DFØ = DF1                                                       INE 151Ø
        DF1 = (-DX*DFØ+.5DØ*DFM1)/DBLE(FLOAT(N))                        INE 152Ø
        F(N) = SNGL(DF1)                                                INE 153Ø
   5Ø CONTINUE                                                          INE 154Ø
      RETURN                                                            INE 155Ø
C                                                                       INE 156Ø
C EVALUATION OF INERFC(X),X.LT.Ø.,BY FORWARD RECURSION                  INE 157Ø
C                                                                       INE 158Ø
   6Ø F1 = SNGL(DSUM)                                                   INE 159Ø
   7Ø FZERO = F1                                                        INE 16ØØ
      DO 8Ø N=1,NMAX                                                    INE 161Ø
        FM1 = FØ                                                        INE 162Ø
        FØ = F1                                                         INE 163Ø
        F1 = (-X*FØ+.5*FM1)/FLOAT(N)                                    INE 164Ø
        F(N) = F1                                                       INE 165Ø
   8Ø CONTINUE                                                          INE 166Ø
      RETURN                                                            INE 167Ø
   9Ø IF (X.LT.Ø.) GO TO 11Ø                                            INE 168Ø
  1ØØ NØ = NMAX                                                         INE 169Ø
      XØ = X                                                            INE 17ØØ
      GO TO 13Ø                                                         INE 171Ø
  11Ø IF (XSQ.LT.(ACC+1.)*AL1Ø-.572) GO TO 12Ø                          INE 172Ø
      FØ = C*S                                                          INE 173Ø
      F1 = 2.                                                           INE 174Ø
      GO TO 7Ø                                                          INE 175Ø
  12Ø I = 2                                                             INE 176Ø
      NØ = 1                                                            INE 177Ø
      XØ = -X                                                           INE 178Ø
C                                                                       INE 179Ø
C BACKWARD RECURRENCE ALGORITHM FOR EVALUATING FN(ABS(X)) FOR N=1,2,    INE 18ØØ
C ...,NØ, WHERE NØ=NMAX IF X.GT.Ø. AND NØ=1 IF X.LT.Ø.                  INE 181Ø
C                                                                       INE 182Ø
  13Ø DO 14Ø N=1,NØ                                                     INE 183Ø
        R1(N) = Ø.                                                      INE 184Ø
```

```
   14Ø CONTINUE                                                          INE 185Ø
       FNØ = FLOAT(NØ)                                                   INE 186Ø
       NU = IFIX((SQRT(FNØ)+(2.3Ø26*ACC+1.3863)/(2.8284*XØ))**2-5.)      INE 187Ø
   15Ø NU = NU + 1Ø                                                      INE 188Ø
       IF (NU.GT.5ØØØ) GO TO 28Ø                                         INE 189Ø
       NØP1 = NØ + 1                                                     INE 19ØØ
       NUM = NU - NØP1                                                   INE 191Ø
       DO 16Ø N=1,NØ                                                     INE 192Ø
         RØ(N) = R1(N)                                                   INE 193Ø
   16Ø CONTINUE                                                          INE 194Ø
       R = Ø.                                                            INE 195Ø
       DO 17Ø K=1,NUM                                                    INE 196Ø
         N = NU - K                                                      INE 197Ø
         R = .5/(XØ+FLOAT(N+1)*R)                                        INE 198Ø
   17Ø CONTINUE                                                          INE 199Ø
       DO 18Ø K=1,NØ                                                     INE 2ØØØ
         N = NØP1 - K                                                    INE 2Ø1Ø
         R = .5/(XØ+FLOAT(N+1)*R)                                        INE 2Ø2Ø
         R1(N) = R                                                       INE 2Ø3Ø
   18Ø CONTINUE                                                          INE 2Ø4Ø
       DO 19Ø N=1,NØ                                                     INE 2Ø5Ø
         IF (ABS(R1(N)-RØ(N)).GT.EPS*ABS(R1(N))) GO TO 15Ø               INE 2Ø6Ø
   19Ø CONTINUE                                                          INE 2Ø7Ø
       FØ = .5*C/(XØ+R1(1))                                              INE 2Ø8Ø
       FZERO = FØ                                                        INE 2Ø9Ø
       FF = FØ                                                           INE 21ØØ
       DO 2ØØ N=1,NØ                                                     INE 211Ø
         FF = R1(N)*FF                                                   INE 212Ø
         F(N) = FF                                                       INE 213Ø
   2ØØ CONTINUE                                                          INE 214Ø
       GO TO (21Ø, 22Ø), I                                               INE 215Ø
   21Ø RETURN                                                            INE 216Ø
C                                                                        INE 217Ø
C STARTING VALUE FOR INERFC(X),X.LT.Ø.,PRIOR TO FORWARD RECURSION        INE 218Ø
C                                                                        INE 219Ø
   22Ø F1 = 2. - S*FØ                                                    INE 22ØØ
       FØ = C*S                                                          INE 221Ø
       GO TO 7Ø                                                          INE 222Ø
   23Ø IFLAG = 1                                                         INE 223Ø
       RETURN                                                            INE 224Ø
   24Ø IFLAG = 2                                                         INE 225Ø
       RETURN                                                            INE 226Ø
   25Ø IFLAG = 3                                                         INE 227Ø
       RETURN                                                            INE 228Ø
   26Ø IFLAG = 4                                                         INE 229Ø
       RETURN                                                            INE 23ØØ
   27Ø IFLAG = 5                                                         INE 231Ø
       RETURN                                                            INE 232Ø
   28Ø IFLAG = 6                                                         INE 233Ø
       RETURN                                                            INE 234Ø
       END                                                               INE 235Ø
```

ALGORITHM 522
ESOLVE, Congruence Techniques for the Exact Solution of Integer Systems of Linear Equations [F4]

S. CABAY and T. P. L. LAM
University of Alberta

Key Words and Phrases: symbolic and algebraic manipulation, systems of linear equations, congruence techniques, modular methods, Chinese remainder theorem
CR Categories: 5.14
Language: Fortran

DESCRIPTION

Descriptions of the main algorithm, ESOLVE, and accompanying subroutines SUBBND, MRADIX, and FRADIX, together with experimental results, are given in [1].

REFERENCES

1. CABAY, S., AND LAM, T.P.L. Congruence techniques for the exact solution of integer systems of linear equations. *ACM Trans. Math. Software 3*, 4 (Dec. 1977), 386–497.

ALGORITHM

```
C                                                                      MAINØØ1Ø
C**********************************************************************MAINØØ2Ø
C                                                                     *MAINØØ3Ø
C                       TWO TEST EXAMPLES                             *MAINØØ4Ø
C                       -----------------                             *MAINØØ5Ø
C                                                                     *MAINØØ6Ø
C**********************************************************************MAINØØ7Ø
C                                                                      MAINØØ8Ø
      INTEGER P(1ØØ),AMOD(2Ø,21),Y(32,2Ø,1),DET(32),T,TWOT1,           MAINØØ9Ø
     1        SUM(7),GAMMA(7),DELTA(7),A(1Ø),C(1Ø),B                   MAINØ1ØØ
      INTEGER A1(1,5,5),B1(1,5,1),A2(3,2Ø,2Ø),B2(3,2Ø,1),TEMP1,Q       MAINØ11Ø
      REAL S(21)                                                       MAINØ12Ø
      COMMON /PRIMEB/ P, IPRIME(1ØØ)                                   MAIN 13Ø
      B = 1ØØØØ                                                        MAINØ14Ø
C                                                                      MAINØ15Ø
C**********************************************************************MAINØ16Ø
C                                                                     *MAINØ17Ø
C                 EXAMPLE 1 - PASCAL MATRIX.                          *MAINØ18Ø
C                                                                     *MAINØ19Ø
```

Received 6 June 1977.

This work was supported by the National Research Council of Canada under Grant NRC A8316.
Author's address: Department of Computing Science, University of Alberta, Edmonton, Canada T6G 2E1.

```
C                     A1*X = B1.                                *        MAINØ2ØØ
C                                                               *        MAINØ21Ø
C                       T = 1                                   *        MAINØ22Ø
C                       N = 5                                   *        MAINØ23Ø
C                       M = 1                                   *        MAINØ24Ø
C                                                               *        MAINØ25Ø
C***************************************************************        MAINØ26Ø
C                                                                        MAINØ27Ø
      T = 1                                                              MAINØ28Ø
      N = 5                                                              MAINØ29Ø
      M = 1                                                              MAINØ3ØØ
      NM = N+M                                                           MAINØ31Ø
C                                                                        MAINØ32Ø
C     **GENERATE THE MATRICES A1 AND B1.                                 MAINØ33Ø
C                                                                        MAINØ34Ø
      DO 1 I=1,N                                                         MAINØ35Ø
         A1(1,I,1) = 1                                                   MAINØ36Ø
 1       A1(1,1,I) = 1                                                   MAINØ37Ø
C     OD                                                                 MAINØ38Ø
      DO 2 I=2,N                                                         MAINØ39Ø
         DO 2 J=2,N                                                      MAINØ4ØØ
 2          A1(1,I,J) = A1(1,I,J-1)+A1(1,I-1,J)                         MAINØ41Ø
C        OD                                                              MAINØ42Ø
C     OD                                                                 MAINØ43Ø
      B1(1,1,1) = 1                                                      MAINØ44Ø
      DO 3 I=2,N                                                         MAINØ45Ø
 3       B1(1,I,1) = B1(1,I-1,1)+A1(1,I,N)                               MAINØ46Ø
C     OD                                                                 MAINØ47Ø
C                                                                        MAINØ48Ø
C     **PRINT A1 AND B1 IN FIXED-RADIX FORM.                             MAINØ49Ø
C                                                                        MAINØ5ØØ
      WRITE(6,42)                                                        MAINØ51Ø
      DO 5 K=1,T                                                         MAINØ52Ø
         WRITE(6,46) K                                                   MAINØ53Ø
         DO 4 I=1,N                                                      MAINØ54Ø
 4          WRITE(6,43) (A1(K,I,J),J=1,N)                               MAINØ55Ø
C        OD                                                              MAINØ56Ø
 5                                                                  CONTINUEMAINØ57Ø
C     OD                                                                 MAINØ58Ø
      WRITE(6,44)                                                        MAINØ59Ø
      DO 7 K=1,T                                                         MAINØ6ØØ
         WRITE(6,49) K                                                   MAINØ61Ø
         DO 6 I=1,N                                                      MAINØ62Ø
 6          WRITE(6,43) (B1(K,I,J),J=1,M)                               MAINØ63Ø
C        OD                                                              MAINØ64Ø
 7                                                                  CONTINUEMAINØ65Ø
C     OD                                                                 MAINØ66Ø
C                                                                        MAINØ67Ø
C     **CONVERT A1 AND B1 FROM FIXED-RADIX TO MIXED-RADIX FORM.          MAINØ68Ø
C                                                                        MAINØ69Ø
      N1 = 1                                                             MAINØ7ØØ
      TEMP = (FLOAT(N1)*ALOG(FLOAT(B)) + ALOG(2.Ø))/ALOG(FLOAT(P(1))) MAINØ71Ø
      LMAX = TEMP                                                        MAINØ72Ø
      IF (TEMP-FLOAT(LMAX).GT.Ø.Ø1) LMAX = LMAX+1                        MAINØ73Ø
      DO 1Ø I=1,N                                                        MAINØ74Ø
         DO 1Ø J=1,N                                                     MAINØ75Ø
            DO 8 K=1,N1                                                  MAINØ76Ø
 8             A(K) = A1(K,I,J)                                          MAINØ77Ø
C           OD                                                           MAINØ78Ø
            CALL MRADIX(A,N1,C,LMAX,L,B,IER)                             MAINØ79Ø
            DO 9 K=1,L                                                   MAINØ8ØØ
 9             A1(K,I,J) = C(K)                                          MAINØ81Ø
C           OD                                                           MAINØ82Ø
C        OD                                                              MAINØ83Ø
C     OD                                                                 MAINØ84Ø
 1Ø                                                                 CONTINUEMAINØ85Ø
      DO 13 I=1,N                                                        MAINØ86Ø
         DO 13 J=1,M                                                     MAINØ87Ø
            DO 11 K=1,N1                                                 MAINØ88Ø
 11             A(K) = B1(K,I,J)                                         MAINØ89Ø
C           OD                                                           MAINØ9ØØ
            CALL MRADIX(A,N1,C,LMAX,L,B,IER)                             MAINØ91Ø
            DO 12 K=1,L                                                  MAINØ92Ø
 12             B1(K,I,J) = C(K)                                         MAINØ93Ø
C           OD                                                           MAINØ94Ø
```

```
C          OD                                                    MAINØ95Ø
C       OD                                                       MAINØ96Ø
 13                                                          CONTINUEMAINØ97Ø
C                                                                MAINØ98Ø
C     **PRINT A1 AND B1 IN MIXED-RADIX FORM.                     MAINØ99Ø
C                                                                MAIN1ØØØ
        WRITE(6,45)                                              MAIN1Ø1Ø
        DO 15 K=1,T                                              MAIN1Ø2Ø
           WRITE(6,46) K                                         MAIN1Ø3Ø
           DO 14 I=1,N                                           MAIN1Ø4Ø
 14           WRITE(6,47) (A1(K,I,J),J=1,N)                      MAIN1Ø5Ø
C          OD                                                    MAIN1Ø6Ø
 15                                                          CONTINUEMAIN1Ø7Ø
C       OD                                                       MAIN1Ø8Ø
        WRITE(6,48)                                              MAIN1Ø9Ø
        DO 17 K=1,T                                              MAIN11ØØ
           WRITE(6,49) K                                         MAIN111Ø
           DO 16 I=1,N                                           MAIN112Ø
 16           WRITE(6,47) (B1(K,I,J),J=1,M)                      MAIN113Ø
C          OD                                                    MAIN114Ø
 17                                                          CONTINUEMAIN115Ø
C       OD                                                       MAIN116Ø
C                                                                MAIN117Ø
C     **COMPUTE MAXPRM.                                          MAIN118Ø
C                                                                MAIN119Ø
        TWOT1 = 2*T+1                                            MAIN12ØØ
        CALL SUBBND(A1,B1,GAMMA,DELTA,S,SUM,T,N,M,NM,            MAIN121Ø
     1              TWOT1,BOUND,MAXPRM)                          MAIN122Ø
        WRITE(6,5Ø) BOUND,MAXPRM                                 MAIN123Ø
C                                                                MAIN124Ø
C     **SOLVE THE SYSTEM OF LINEAR EQUATIONS A1*X=B1.            MAIN125Ø
C                                                                MAIN126Ø
        CALL DRIVER(A1,B1,AMOD,T,N,M,NM,Y,DET,MAXPRM,B)          MAIN127Ø
C                                                                MAIN128Ø
C*****************************************************************MAIN129Ø
C                                                               *  MAIN13ØØ
C                  EXAMPLE 2 - PASCAL MATRIX.                   *  MAIN131Ø
C                                                               *  MAIN132Ø
C                      A2*X = B2.                               *  MAIN133Ø
C                                                               *  MAIN134Ø
C                        T = 3                                  *  MAIN135Ø
C                        N = 2Ø                                 *  MAIN136Ø
C                        M = 1                                  *  MAIN137Ø
C                                                               *  MAIN138Ø
C*****************************************************************MAIN139Ø
C                                                                MAIN14ØØ
        T = 3                                                    MAIN141Ø
        N = 2Ø                                                   MAIN142Ø
        M = 1                                                    MAIN143Ø
        NM = N+M                                                 MAIN144Ø
C                                                                MAIN145Ø
C     **GENERATE THE MATRICES A2 AND B2.                         MAIN146Ø
C                                                                MAIN147Ø
        DO 18 I=1,N                                              MAIN148Ø
           A2(3,I,1) = 1                                         MAIN149Ø
 18        A2(3,1,I) = 1                                         MAIN15ØØ
C       OD                                                       MAIN151Ø
        DO 19 K=1,2                                              MAIN152Ø
           DO 19 I=1,N                                           MAIN153Ø
              A2(K,I,1) = Ø                                      MAIN154Ø
 19           A2(K,1,I) = Ø                                      MAIN155Ø
C          OD                                                    MAIN156Ø
C       OD                                                       MAIN157Ø
        DO 21 I=2,N                                              MAIN158Ø
           DO 21 J=2,N                                           MAIN159Ø
              Q = Ø                                              MAIN16ØØ
              TEMP1 = Ø                                          MAIN161Ø
              DO 2Ø K=1,T                                        MAIN162Ø
                 K1 = T-K+1                                      MAIN163Ø
                 TEMP1 = A2(K1,I,J-1)+A2(K1,I-1,J)+Q             MAIN164Ø
                 Q = TEMP1/B                                     MAIN165Ø
 2Ø              A2(K1,I,J) = TEMP1 - Q*B                        MAIN166Ø
C             OD                                                 MAIN167Ø
C          OD                                                    MAIN168Ø
C       OD                                                       MAIN169Ø
```

```
 21                                                                  CONTINUEMAIN17ØØ
       B2(1,1,1) = Ø                                                        MAIN171Ø
       B2(2,1,1) = Ø                                                        MAIN172Ø
       B2(3,1,1) = 1                                                        MAIN173Ø
       DO 23 I=2,N                                                          MAIN174Ø
          Q = Ø                                                             MAIN175Ø
          TEMP1 = Ø                                                         MAIN176Ø
          DO 22 K=1,T                                                       MAIN177Ø
             K1 = T-K+1                                                     MAIN178Ø
             TEMP1 = B2(K1,I-1,1)+A2(K1,I,N)+Q                              MAIN179Ø
             Q = TEMP1/B                                                    MAIN18ØØ
 22          B2(K1,I,1) = TEMP1 - Q*B                                       MAIN181Ø
C         OD                                                               MAIN182Ø
C      OD                                                                  MAIN183Ø
 23                                                                  CONTINUEMAIN184Ø
C                                                                           MAIN185Ø
C     **PRINT A2 AND B2 IN FIXED-RADIX FORM.                                MAIN186Ø
C                                                                           MAIN187Ø
       WRITE(6,42)                                                          MAIN188Ø
       DO 25 K=1,T                                                          MAIN189Ø
          WRITE(6,46) K                                                     MAIN19ØØ
          K1 = T-K+1                                                        MAIN191Ø
          DO 24 I=1,N                                                       MAIN192Ø
 24          WRITE(6,43) (A2(K1,I,J),J=1,N)                                 MAIN193Ø
C         OD                                                               MAIN194Ø
C      OD                                                                  MAIN195Ø
 25                                                                  CONTINUEMAIN196Ø
       WRITE(6,44)                                                          MAIN197Ø
       DO 27 K=1,T                                                          MAIN198Ø
          WRITE(6,49) K                                                     MAIN199Ø
          K1 = T-K+1                                                        MAIN2ØØØ
          DO 26 I=1,N                                                       MAIN2Ø1Ø
 26          WRITE(6,43) (B2(K1,I,J),J=1,M)                                 MAIN2Ø2Ø
C         OD                                                               MAIN2Ø3Ø
C      OD                                                                  MAIN2Ø4Ø
 27                                                                  CONTINUEMAIN2Ø5Ø
C                                                                           MAIN2Ø6Ø
C     **CONVERT A2 AND B2 FROM FIXED-RADIX TO MIXED-RADIX FORM.             MAIN2Ø7Ø
C                                                                           MAIN2Ø8Ø
       N1 = 3                                                               MAIN2Ø9Ø
       TEMP = (FLOAT(N1)*ALOG(FLOAT(B)) + ALOG(2.Ø))/ALOG(FLOAT(P(1))) MAIN21ØØ
       LMAX = TEMP                                                          MAIN211Ø
       IF (TEMP-FLOAT(LMAX).GT.Ø.Ø1) LMAX = LMAX+1                          MAIN212Ø
       DO 32 I=1,N                                                          MAIN213Ø
          DO 32 J=1,N                                                       MAIN214Ø
             DO 28 K=1,N1                                                   MAIN215Ø
 28             A(K) = A2(K,I,J)                                           MAIN216Ø
C            OD                                                           MAIN217Ø
             CALL MRADIX(A,N1,C,LMAX,L,B,IER)                               MAIN218Ø
             DO 29 K=1,L                                                    MAIN219Ø
 29             A2(K,I,J) = C(K)                                           MAIN22ØØ
C            OD                                                           MAIN221Ø
             IF (L.LT.N1)                                           GOTO 3Ø MAIN222Ø
                                                                    GOTO 32 MAIN223Ø
C            THEN                                                         MAIN224Ø
 3Ø             L1 = L+1                                                   MAIN225Ø
                DO 31 K=L1,N1                                           MAIN226Ø
 31                A2(K,I,J) = Ø                                         MAIN227Ø
C               OD                                                       MAIN228Ø
C         OD                                                             MAIN229Ø
C      OD                                                                MAIN23ØØ
 32                                                                  CONTINUEMAIN231Ø
       DO 37 I=1,N                                                          MAIN232Ø
          DO 37 J=1,M                                                       MAIN233Ø
             DO 33 K=1,N1                                                   MAIN234Ø
 33             A(K) = B2(K,I,J)                                           MAIN235Ø
C            OD                                                           MAIN236Ø
             CALL MRADIX(A,N1,C,LMAX,L,B,IER)                               MAIN237Ø
             DO 34 K=1,L                                                    MAIN238Ø
 34             B2(K,I,J) = C(K)                                           MAIN239Ø
C            OD                                                           MAIN24ØØ
             IF (L.LT.N1)                                           GOTO 35 MAIN241Ø
                                                                    GOTO 37 MAIN242Ø
C            THEN                                                         MAIN243Ø
 35             L1 = L+1                                                   MAIN244Ø
```

```
                DO 36 K=L1,N1                                          MAIN245Ø
 36                 B2(K,I,J) = Ø                                      MAIN246Ø
C                   OD                                                 MAIN247Ø
C            OD                                                        MAIN248Ø
C         OD                                                           MAIN249Ø
 37                                                                  CONTINUEMAIN25ØØ
C                                                                      MAIN251Ø
C       **PRINT A2 AND B2 IN MIXED-RADIX FORM.                         MAIN252Ø
C                                                                      MAIN253Ø
          WRITE(6,45)                                                  MAIN254Ø
          DO 39 K=1,T                                                  MAIN255Ø
             WRITE(6,46) K                                             MAIN256Ø
             DO 38 I=1,N                                               MAIN257Ø
 38             WRITE(6,47) (A2(K,I,J),J=1,N)                         MAIN258Ø
C            OD                                                        MAIN259Ø
 39                                                                  CONTINUEMAIN26ØØ
C         OD                                                           MAIN261Ø
          WRITE(6,48)                                                  MAIN262Ø
          DO 41 K=1,T                                                  MAIN263Ø
             WRITE(6,49) K                                             MAIN264Ø
             DO 4Ø I=1,N                                               MAIN265Ø
 4Ø             WRITE(6,47) (B2(K,I,J),J=1,M)                         MAIN266Ø
C            OD                                                        MAIN267Ø
 41                                                                  CONTINUEMAIN268Ø
C         OD                                                           MAIN269Ø
C                                                                      MAIN27ØØ
C       **COMPUTE MAXPRM.                                              MAIN271Ø
C                                                                      MAIN272Ø
          TWOT1 = 2*T+1                                                MAIN273Ø
          CALL SUBBND(A2,B2,GAMMA,DELTA,S,SUM,T,N,M,NM,                MAIN274Ø
     1              TWOT1,BOUND,MAXPRM)                                MAIN275Ø
          WRITE(6,5Ø) BOUND,MAXPRM                                     MAIN276Ø
C                                                                      MAIN277Ø
C       **SOLVE THE SYSTEM OF LINEAR EQUATIONS A2*X=B2.                MAIN278Ø
C                                                                      MAIN279Ø
          CALL DRIVER(A2,B2,AMOD,T,N,M,NM,Y,DET,MAXPRM,B)              MAIN28ØØ
C                                                                      MAIN281Ø
 42    FORMAT(1H1,16X,5HINPUT/17X,5(1H*)/5X,                          MAIN282Ø
     1       38HFIXED-RADIX REPRESENTATION, BASE=1ØØØØ/1ØX,          MAIN283Ø
     2       4ØHA = A(1,N,N)+A(2,N,N)*BASE+...+A(T,N,N)*,            MAIN284Ø
     3       11HBASE**(T-1))                                         MAIN285Ø
 43    FORMAT(5X,2Ø(I4,2X))                                           MAIN286Ø
 44    FORMAT(/1ØX,4ØHB = B(1,N,M)+B(2,N,M)*BASE+...+B(T,N,M)*,       MAIN287Ø
     1       11HBASE**(T-1))                                         MAIN288Ø
 45    FORMAT(/5X,26HMIXED-RADIX REPRESENTATION/1ØX,                  MAIN289Ø
     1       4ØHA = A(1,N,N)+A(2,N,N)*P(1)+...+A(T,N,N)*,            MAIN29ØØ
     2       15HP(1)*...*P(T-1))                                     MAIN291Ø
 46    FORMAT(/1ØX,2HA(,I1,5H,N,N))                                   MAIN292Ø
 47    FORMAT(1H ,2Ø(I6))                                             MAIN293Ø
 48    FORMAT(/1ØX,4ØHB = B(1,N,M)+B(2,N,M)*P(1)+...+B(T,N,M)*,       MAIN294Ø
     1       15HP(1)*...*P(T-1))                                     MAIN295Ø
 49    FORMAT(/1ØX,2HB(,I1,5H,N,M))                                   MAIN296Ø
 5Ø    FORMAT(/1ØX,13HLOG(BOUND) = ,F8.4,2X,9HMAXPRM = ,I3)           MAIN297Ø
       STOP                                                           MAIN298Ø
       END                                                            MAIN299Ø


C                                                                      DRIVØØ1Ø
C**********************************************************          DRIVØØ2Ø
C                                                         *            DRIVØØ3Ø
C                       CALL ESOLVE.                      *            DRIVØØ4Ø
C                 PRINT OUTPUTS IN MIXED-RADIX FORM.      *            DRIVØØ5Ø
C                       CALL FRADIX.                      *            DRIVØØ6Ø
C                 PRINT OUTPUTS IN FIXED-RADIX FORM.      *            DRIVØØ7Ø
C                                                         *            DRIVØØ8Ø
C**********************************************************          DRIVØØ9Ø
C                                                                      DRIVØ1ØØ
      SUBROUTINE DRIVER(A,B,AMOD,T,N,M,NM,Y,DET,MAXPRM,BASE)           DRIVØ11Ø
       INTEGER T,A(T,N,N),B(T,N,M),AMOD(N,NM),Y(MAXPRM,N,M),           DRIVØ12Ø
     1        DET(MAXPRM),BASE,P(1ØØ)                                 DRIVØ13Ø
       INTEGER  A1(1Ø),C1(1Ø)                                         DRIV 46Ø
       COMMON /PRIMEB/ P, IPRIME(1ØØ)                                 DRIV 14Ø
       LOGICAL RTEST                                                  DRIVØ15Ø
C                                                                      DRIVØ16Ø
C      **SOLVE THE SYSTEM OF LINEAR EQUATIONS AX=B.                    DRIVØ17Ø
```

```
C                                                                       DRIV0180
      RTEST = .TRUE.                                                    DRIV0190
      CALL ESOLVE(A,B,AMOD,T,N,M,NM,DET,Y,MAXPRM,NOCOEF,                DRIV0200
     1            RTEST,IER)                                            DRIV0210
C                                                                       DRIV0220
C     **PRINT OUTPUT PARAMETERS.                                        DRIV0230
C                                                                       DRIV0240
      WRITE(6,6) IER,NOCOEF                                             DRIV0250
C                                                                       DRIV0260
C     **PRINT Y IN MIXED-RADIX FORM.                                    DRIV0270
C                                                                       DRIV0280
      WRITE(6,7)                                                        DRIV0290
      DO 2 J=1,M                                                        DRIV0300
         WRITE(6,8) J                                                   DRIV0310
         DO 1 I=1,N                                                     DRIV0320
 1          WRITE(6,10) (Y(K,I,J),K=1,NOCOEF)                          DRIV0330
C        OD                                                             DRIV0340
 2                                                                  CONTINUEDRIV0350
C     OD                                                                DRIV0360
C                                                                       DRIV0370
C     **PRINT DET IN MIXED-RADIX FORM.                                  DRIV0380
C                                                                       DRIV0390
      WRITE(6,9)                                                        DRIV0400
      WRITE(6,10) (DET(K),K=1,NOCOEF)                                   DRIV0410
C                                                                       DRIV0420
C     **CONVERT Y AND DET FROM MIXED-RADIX TO FIXED-RADIX FORM.         DRIV0430
C     **PRINT Y AND DET IN FIXED-RADIX FORM.                            DRIV0440
C                                                                       DRIV0450
      TEMP = (FLOAT(NOCOEF)*ALOG(FLOAT(P(NOCOEF))) - ALOG(2.0)) /       DRIV0470
     1       ALOG(FLOAT(BASE))                                          DRIV0480
      LMAX = TEMP                                                       DRIV0490
      IF (TEMP-FLOAT(LMAX).GT.0.01) LMAX = LMAX+1                       DRIV0500
      WRITE(6,11)                                                       DRIV0510
      DO 5 J=1,M                                                        DRIV0520
         WRITE(6,8) J                                                   DRIV0530
         DO 4 I=1,N                                                     DRIV0540
            DO 3 K=1,NOCOEF                                             DRIV0550
 3             C1(K) = Y(K,I,J)                                         DRIV0560
C           OD                                                          DRIV0570
            CALL FRADIX(C1,NOCOEF,A1,LMAX,L,BASE,IER)                   DRIV0580
 4          WRITE(6,10) (A1(II),II=1,L)                                DRIV0590
C        OD                                                             DRIV0600
 5                                                                  CONTINUEDRIV0610
C     OD                                                                DRIV0620
      CALL FRADIX(DET,NOCOEF,A1,LMAX,L,BASE,IER)                        DRIV0630
      WRITE(6,12) (A1(I),I=1,L)                                         DRIV0640
C                                                                       DRIV0650
 6    FORMAT(/16X,7HOUTPUTS/16X,7(1H*)/6X,                              DRIV0660
     1       6HIER = ,I3,5X,9HNOCOEF = ,I3)                             DRIV0670
 7    FORMAT(/16X,21HY IN MIXED-RADIX FORM/10X,                         DRIV0680
     1       35HFROM LEFT TO RIGHT, THE NUMBERS ARE,                    DRIV0690
     2       25H COEF(1),...,COEF(L), AND/10X,                          DRIV0700
     3       33HY(I,J) = COEF(1)+COEF(2)*P(1)+...,                      DRIV0710
     4       23H+COEF(L)*P(1)*...P(L-1))                                DRIV0720
 8    FORMAT(/10X,6HY(I,N,,I1,1H))                                      DRIV 730
 9    FORMAT(/16X,23HDET IN MIXED-RADIX FORM/10X,                       DRIV0740
     1       35HFROM LEFT TO RIGHT, THE NUMBERS ARE,                    DRIV0750
     2       24H COEF(1),...,COEF(L),AND/10X,                           DRIV0760
     3       30HDET = COEF(1)+COEF(2)*P(1)+...,                         DRIV0770
     4       23H+COEF(L)*P(1)*...P(L-1))                                DRIV0780
 10   FORMAT(16X,20(I6))                                                DRIV0790
 11   FORMAT(/16X,21HY IN FIXED-RADIX FORM/10X,                         DRIV0800
     1       35HFROM LEFT TO RIGHT, THE NUMBERS ARE,                    DRIV0810
     2       24H COEF(1),...,COEF(L),AND/10X,                           DRIV0820
     3       37HY(I,J) = COEF(1)*BASE**(L-1)+COEF(2)*,                  DRIV0830
     4       23HBASE**(L-2)+...+COEF(L))                                DRIV0840
 12   FORMAT(/16X,23HDET IN FIXED-RADIX FORM/10X,                       DRIV0850
     1       35HFROM LEFT TO RIGHT, THE NUMBERS ARE,                    DRIV0860
     2       24H COEF(1),...,COEF(L),AND/10X,                           DRIV0870
     3       34HDET = COEF(1)*BASE**(L-1)+COEF(2)*,                     DRIV0880
     4       21HBASE(N-2)+...+COEF(L)/16X,10(I6))                       DRIV0890
      RETURN                                                            DRIV0900
      END                                                               DRIV0910
```

```
      SUBROUTINE ESOLVE(A,B,AMOD,T,N,M,NM,DET,Y,MAXPRM,NOCOEF,          ESOLØØ1Ø
     1                  RTEST,IER)                                      ESOLØØ2Ø
C                                                                       ESOLØØ3Ø
C***************************************************************      ESOLØØ4Ø
C                                                              *        ESOLØØ5Ø
C THIS SUBROUTINE SOLVES EXACTLY THE SYSTEM OF LINEAR EQUATIONS*        ESOLØØ6Ø
C AX=B, WHERE A( ,N,N) AND B( ,N,M) ARE MATRICES WITH MULTIPLE-*        ESOLØØ7Ø
C PRECISION INTEGER COEFFICIENTS EXPRESSED IN MIXED-RADIX FORM.*        ESOLØØ8Ø
C THE SUBROUTINE NORMALLY RETURNS INTEGER VALUES FOR           *        ESOLØØ9Ø
C DET=DETERMINANT(A) AND Y=A(ADJOINT)*B, ALSO IN MIXED-RADIX   *        ESOLØ1ØØ
C FORM. IF THE SOLUTION X IS REQUIRED, THE USER NEED ONLY      *        ESOLØ11Ø
C COMPUTE X=Y/DET.  (FOR CONVERSION OF INTEGERS FROM MIXED-    *        ESOLØ12Ø
C RADIX TO FIXED-RADIX FORMS AND CONVERSELY, SUBROUTINES FRADIX*        ESOLØ13Ø
C AND MRADIX CAN BE USED.)                                     *        ESOLØ14Ø
C                                                              *        ESOLØ15Ø
C  PRIME  IS A LINEAR ARRAY CONTAINING 1ØØ DISTINCT PRIME      *        ESOLØ16Ø
C (INPUT) INTEGERS IN ASCENDING ORDER.  THE PRIMES ARE CHOSEN  *        ESOLØ17Ø
C         AS LARGE AS POSSIBLE SUBJECT TO THE CONDITION THAT   *        ESOLØ18Ø
C         FOR ALL I AND J PRIME(I)*PRIME(J) DOES NOT OVERFLOW  *        ESOLØ19Ø
C         AN INTEGER WORD.  THESE PRIMES ARE USED AS RADII FOR *        ESOLØ2ØØ
C         THE REPRESENTATION OF INTEGERS IN MIXED-RADIX FORM.  *        ESOLØ21Ø
C IPRIME  IS A LINEAR ARRAY OF INTEGERS SUCH THAT FOR EACH K,  *        ESOLØ22Ø
C (INPUT) IPRIME(K)* PRIME(1)*PRIME(2)*...*PRIME(K-1) = 1,     *        ESOLØ23Ø
C                                          MODULO PRIME(K).    *        ESOLØ24Ø
C      A  IS AN INTEGER MATRIX OF DIMENSION T BY N BY N.  THE  *        ESOLØ25Ø
C (INPUT) FIRST DIMENSION CONTAINS THE COEFFICIENTS OF THE     *        ESOLØ26Ø
C         MIXED-RADIX REPRESENTATION OF THE MULTIPLE-PRECISION *        ESOLØ27Ø
C         COMPONENTS OF THE N BY N MATRIX A( ,N,N).  THAT IS,  *        ESOLØ28Ø
C            A( ,I,J) = A(1,I,J)                               *        ESOLØ29Ø
C                     + A(2,I,J)*PRIME(1)                      *        ESOLØ3ØØ
C                     .                                        *        ESOLØ31Ø
C                     .                                        *        ESOLØ32Ø
C                     .                                        *        ESOLØ33Ø
C                     + A(T,I,J)*PRIME(1)*...*PRIME(T-1),      *        ESOLØ34Ø
C         FOR I,J=1,2,...,N.                                   *        ESOLØ35Ø
C      B  IS AN INTEGER MATRIX OF DIMENSION T BY N BY M WITH   *        ESOLØ36Ø
C (INPUT) A SIMILAR NOTATIONAL CORRESPONDENCE MADE FOR A ABOVE.*        ESOLØ37Ø
C   AMOD  IS AN N BY N+M MATRIX USED FOR TEMPORARY STORAGE OF  *        ESOLØ38Ø
C  (TEMP) THE AUGMENTED MATRIX (A,B) MODULO THE VARIOUS PRIMES *        ESOLØ39Ø
C         PRIME(K).  THIS MATRIX IS INCLUDED IN THE ARGUMENT   *        ESOLØ4ØØ
C         LIST ONLY TO PERMIT ITS DIMENSIONS TO BE VARIABLE.   *        ESOLØ41Ø
C      T  IS THE NUMBER OF RADII, PRIME(1),...,PRIME(T), USED  *        ESOLØ42Ø
C (INPUT) TO REPRESENT EACH COMPONENT OF A( ,N,N) AND B( ,N,M) *        ESOLØ43Ø
C         IN MIXED-RADIX FORM.                                 *        ESOLØ44Ø
C      N  IS THE NUMBER OF EQUATIONS AND THE NUMBER OF UNKNOWNS*        ESOLØ45Ø
C (INPUT) IN THE SYSTEM.  (I.E., N IS THE SIZE OF THE SECOND   *        ESOLØ46Ø
C         AND THIRD DIMENSIONS OF A.)                          *        ESOLØ47Ø
C      M  IS THE NUMBER OF RIGHT-HAND VECTORS FOR WHICH THE    *        ESOLØ48Ø
C (INPUT) SYSTEM IS TO BE SOLVED.  (I.E., M IS THE SIZE OF THE *        ESOLØ49Ø
C         THIRD DIMENSION OF B.)                               *        ESOLØ5ØØ
C     NM  IS EQUAL TO N+M.                                     *        ESOLØ51Ø
C (INPUT)                                                      *        ESOLØ52Ø
C    DET  IS A VECTOR OF DIMENSION MAXPRM WHICH CONTAINS THE   *        ESOLØ53Ø
C(OUTPUT) COEFFICIENTS OF THE MIXED-RADIX REPRESENTATION OF    *        ESOLØ54Ø
C         DETERMINANT(A)                                       *        ESOLØ55Ø
C            = DET(1)                                          *        ESOLØ56Ø
C            + DET(2)*PRIME(1)                                 *        ESOLØ57Ø
C            .                                                 *        ESOLØ58Ø
C            .                                                 *        ESOLØ59Ø
C            .                                                 *        ESOLØ6ØØ
C            + DET(MAXPRM)*PRIME(1)*...*PRIME(MAXPRM-1).       *        ESOLØ61Ø
C      Y  IS A MATRIX OF DIMENSION MAXPRM BY N BY M.  THE FIRST*        ESOLØ62Ø
C(OUTPUT) DIMENSION CONTAINS THE COEFFICIENTS OF THE MIXED-    *        ESOLØ63Ø
C         RADIX REPRESENTATION OF Y( ,N,M) = A(ADJOINT)*B:     *        ESOLØ64Ø
C            Y( ,I,J)                                          *        ESOLØ65Ø
C               = Y(1,I,J)                                     *        ESOLØ66Ø
C               + Y(2,I,J)*PRIME(1)                            *        ESOLØ67Ø
C               .                                              *        ESOLØ68Ø
C               .                                              *        ESOLØ69Ø
C               .                                              *        ESOLØ7ØØ
C               + Y(MAXPRM,I,J)*PRIME(1)*...*PRIME(MAXPRM-1).  *        ESOLØ71Ø
C MAXPRM  IS THE MAXIMUM NUMBER OF RADII, PRIME(1),...,        *        ESOLØ72Ø
C (INPUT) PRIME(MAXPRM), REQUIRED TO REPRESENT DETERMINANT(A)  *        ESOLØ73Ø
C         AND A(ADJOINT)*B IN MIXED-RADIX FORM.  MAXPRM SHOULD *        ESOLØ74Ø
C         BE CHOSEN SO THAT                                    *        ESOLØ75Ø
C            ABS(DETERMINANT(A)), ABS(Y( ,I,J))                *        ESOLØ76Ø
```

```
C                < PRIME(1)*...*PRIME(MAXPRM)/2, I=1,2,...,N,   *        ESOLØ77Ø
C                                                J=1,2,...,M.   *        ESOLØ78Ø
C          CHOOSING MAXPRM = N*T + N*LOG(N)/(2*LOG(PRIME(1)))   *        ESOLØ79Ø
C          WILL SUFFICE. (A TIGHTER BOUND CAN BE OBTAINED USING *        ESOLØ8ØØ
C          SUBROUTINE SUBBND.) MAXPRM MUST BE LESS THAN 1Ø1.    *        ESOLØ81Ø
C NOCOEF   IS THE NUMBER OF RADII ACTUALLY USED TO REPRESENT    *        ESOLØ82Ø
C(OUTPUT) DET AND Y( ,I,J) IN MIXED-RADIX FORM.                 *        ESOLØ83Ø
C          NOCOEF <= MAXPRM.                                    *        ESOLØ84Ø
C  RTEST   IF RTEST = .FALSE. , MAXPRM COEFFICIENTS IN THE      *        ESOLØ85Ø
C (INPUT)     MIXED-RADIX REPRESENTATION OF DET AND Y( ,N,M)    *        ESOLØ86Ø
C             ARE COMPUTED.                                     *        ESOLØ87Ø
C          IF RTEST = .TRUE. , THE ALGORITHM CONTINUES TO       *        ESOLØ88Ø
C             ITERATE UNTIL TR  (USUALLY TR=T, BUT SOMETIMES    *        ESOLØ89Ø
C             TR=T+1)  SUCCESSIVE ZERO COEFFICIENTS IN THE      *        ESOLØ9ØØ
C             MIXED-RADIX REPRESENTATION OF DET AND Y( ,N,M)    *        ESOLØ91Ø
C             ARE ENCOUNTERED, THAT IS, WHENEVER FOR            *        ESOLØ92Ø
C             K = NOCOEF+1,...,NOCOEF+TR  (NOCOEF+TR <= MAXPRM),*        ESOLØ93Ø
C                DET(K) = Ø                                     *        ESOLØ94Ø
C                Y(K,I,J) = Ø, I=1,2,...,N, J=1,2,...,M.       *        ESOLØ95Ø
C             IF NOCOEF > Ø, THE PROGRAM TERMINATES WITH THE    *        ESOLØ96Ø
C                KNOWLEDGE THAT                                 *        ESOLØ97Ø
C                (1) A IS SINGULAR IF DET=Ø, OR                 *        ESOLØ98Ø
C                (2) Y( ,N,M)/DET IS THE SOLUTION TO AX=B IF    *        ESOLØ99Ø
C                    DET¬=Ø. (*WARNING* IN THIS CASE, A COMMON  *        ESOL1ØØØ
C                    FACTOR IN Y( ,N,M) AND DET MAY HAVE        *        ESOL1Ø1Ø
C                    BEEN IGNORED.  THAT IS, DET MIGHT NO       *        ESOL1Ø2Ø
C                    LONGER BE THE DETERMINANT OF A.)           *        ESOL1Ø3Ø
C             IF NOCOEF = Ø, A IS PROBABLY SINGULAR, BUT THE    *        ESOL1Ø4Ø
C                PROGRAM CONTINUES TO ITERATE UNTIL MAXPRM      *        ESOL1Ø5Ø
C                COEFFICIENTS OF DET AND Y( ,N,M) HAVE BEEN     *        ESOL1Ø6Ø
C                COMPUTED.                                      *        ESOL1Ø7Ø
C    IER   IS AN ERROR CODE WHICH IS                            *        ESOL1Ø8Ø
C(OUTPUT) Ø  IF Y( ,N,M)/DET IS THE SOLUTION TO AX=B.           *        ESOL1Ø9Ø
C             (*WARNING*  DET MIGHT NOT BE THE DETERMINANT OF A)*        ESOL11ØØ
C          1  IF MAXPRM COEFFICIENTS IN THE MIXED-RADIX        *        ESOL111Ø
C             REPRESENTATION OF DET AND Y( ,N,M) HAVE BEEN     *        ESOL112Ø
C             COMPUTED.  SINCE THE USER CHOOSES THE VALUE OF   *        ESOL113Ø
C             MAXPRM, THE PROGRAM CANNOT GUANENTEE THAT        *        ESOL114Ø
C             Y( ,N,M)/DET IS THE SOLUTION TO AX=B.            *        ESOL115Ø
C          2  IF A IS SINGULAR.                                *        ESOL116Ø
C          3  IF A IS SINGULAR, OR DET AND Y( ,N,M) ARE NON-ZERO*        ESOL117Ø
C             MULTIPLES OF PRIME(1)*...*PRIME(MAXPRM).         *        ESOL118Ø
C          4  IF THE INPUT PARAMETERS ARE INCORRECT.           *        ESOL119Ø
C                                                             *        ESOL12ØØ
C*************************************************************        ESOL121Ø
C                                                                      ESOL122Ø
      INTEGER T,TR,A(T,N,N),B(T,N,M),DET(MAXPRM),Y(MAXPRM,N,M),        ESOL123Ø
     1        AMOD(N,NM)                                               ESOL124Ø
      LOGICAL RTEST                                                    ESOL125Ø
      INTEGER PT,TK,TT,TT1,DMOD,V1,V3,U1,U3,T1,T3,Q                    ESOL126Ø
      INTEGER PRIME(1ØØ),IPRIME(1ØØ),P,P1,P2                           ESOL127Ø
      LOGICAL TEST                                                     ESOL128Ø
      COMMON /PRIMEB/PRIME,IPRIME                                      ESOL129Ø
      IER=4                                                            ESOL13ØØ
      IF (MAXPRM.GT.1ØØ .OR. N.LT.2 .OR. M.LT.1) RETURN                ESOL131Ø
      IF (N+M.NE.NM) RETURN                                            ESOL132Ø
      N1 = N+1                                                         ESOL133Ø
C                                                                      ESOL134Ø
C     PROCEDURE ZERO-CRITERION **********************************      ESOL135Ø
C        DETERMINED IN THIS PROCEDURE IS THE NUMBER TR OF      *       ESOL136Ø
C        CONSECUTIVE ZERO COEFFICIENTS IN THE MIXED-RADIX      *       ESOL137Ø
C        REPRESENTATION OF DET AND Y REQUIRED TO GUARANTEE THAT*       ESOL138Ø
C        X=Y/DET IS THE SOLUTION OF AX=B. TR=T IF NORM INFINITY*       ESOL139Ø
C        OF THE MATRIX A AUGMENTED BY ANY COLUMN VECTOR OF B IS*       ESOL14ØØ
C        BOUNDED BY 2*PRIME(1)*...*PRIME(T); OTHERWISE, TR=T+1.*       ESOL141Ø
C                                                              *       ESOL142Ø
      TR = T                                                           ESOL143Ø
      IF (RTEST)                                          GOTO 1       ESOL144Ø
                                                          GOTO 6       ESOL145Ø
C     THEN                                                             ESOL146Ø
  1      PT = 2*PRIME(T)                                               ESOL147Ø
         I = Ø                                                         ESOL148Ø
C        REPEAT ... FOR EACH ROW I                                     ESOL149Ø
  2         I = I+1                                                    ESOL15ØØ
            IF (I.LE.N .AND. TR.EQ.T)                     GOTO 3       ESOL151Ø
                                                          GOTO 6       ESOL152Ø
```

```
C              THEN                                                    ESOL153Ø
  3               NORM = Ø                                             ESOL154Ø
                  DO 4 J=1,N                                           ESOL155Ø
  4                   NORM = NORM + IABS(A(T,I,J))                     ESOL156Ø
C                 OD                                                   ESOL157Ø
                  MAXB = Ø                                             ESOL158Ø
                  DO 5 J=1,M                                           ESOL159Ø
  5                  IF (MAXB.LT.IABS(B(T,I,J)))                       ESOL16ØØ
    1                    MAXB = IABS(B(T,I,J))                        ESOL161Ø
C                 OD                                                   ESOL162Ø
                  NORM = NORM + MAXB                                   ESOL163Ø
                  IF (T.GT.1) NORM = NORM + N1                         ESOL164Ø
C                 **HERE, USED IS THE FACT THAT FOR T≠1                ESOL165Ø
C                 **(ABS(K(T)+1)*PRIME(1)*...*PRIME(T-1)) IS           ESOL166Ø
C                 **A BOUND FOR ANY INTEGER GIVEN BY                   ESOL167Ø
C                 **K(1)+K(2)*PRIME(1)+...+K(T)*PRIME(1)*              ESOL168Ø
C                 **...*PRIME(T-1).                                    ESOL169Ø
                  IF (NORM.GT.PT) TR = T+1                             ESOL17ØØ
C             CONTINUE                                                 ESOL171Ø
                                                              GOTO 2     ESOL172Ø
C                                                            *          ESOL173Ø
C       END ZERO-CRITERION ************************************        ESOL174Ø
C                                                                        ESOL175Ø
  6   NOZERO = Ø                                                         ESOL176Ø
      DO 64 ITER=1,MAXPRM                                              ESOL177Ø
C        **THE SYSTEM AX=B IS SOLVED MODULO PRIME(ITER) FOR ALL         ESOL178Ø
C        **ITER = 1,2,...,MAXPRM, OR IF RTEST IS TRUE UNTIL            ESOL179Ø
C        **THE NUMBER OF CONSECUTIVE ZERO COEFFICIENTS (NOZERO)         ESOL18ØØ
C        **IN THE MIXED-RADIX REPRESENTATION OF DET AND Y IS            ESOL181Ø
C        **EQUAL TO TR.                                                 ESOL182Ø
         P = PRIME(ITER)                                                 ESOL183Ø
         P1 = P-1                                                        ESOL184Ø
         P2 = P1/2 + 1                                                   ESOL185Ø
         IP = IPRIME(ITER)                                               ESOL186Ø
C                                                                        ESOL187Ø
C        PROCEDURE MAP *****************************************        ESOL188Ø
C           THIS PROCEDURE COMPUTES AMOD, THE AUGMENTED          *      ESOL189Ø
C           N BY N+M MATRIX (A,B) MODULO P.  THAT IS, FOR        *      ESOL19ØØ
C           I=1,2,...,N, HORNER'S RULE AND MODULO P ARITHMETIC   *      ESOL191Ø
C           IS USED TO COMPUTE                                   *      ESOL192Ø
C             AMOD(I,J) = A(1,I,J) + A(2,I,J)*PRIME(1) +         *      ESOL193Ø
C                         ... + A(T,I,J)*PRIME(1)*...*PRIME(T-1),*     ESOL194Ø
C                                                  J=1,2,...,N,*       ESOL195Ø
C           AMOD(I,J+N) = B(1,I,J) + B(2,I,J)*PRIME(1) +         *      ESOL196Ø
C                         ... + B(T,I,J)*PRIME(1)*...*PRIME(T-1),*     ESOL197Ø
C                                                  J=1,2,...,M.*       ESOL198Ø
C                                                              *        ESOL199Ø
            TT = MINØ(T,ITER)                                            ESOL2ØØØ
            TT1 = TT-1                                                   ESOL2Ø1Ø
            IF (TT.EQ.1)                                      GOTO 7     ESOL2Ø2Ø
                                                              GOTO 11    ESOL2Ø3Ø
C           THEN                                                         ESOL2Ø4Ø
  7            DO 1Ø I=1,N                                             ESOL2Ø5Ø
                  DO 8 J=1,N                                           ESOL2Ø6Ø
  8                  AMOD(I,J) = A(1,I,J)                              ESOL2Ø7Ø
C                 OD                                                   ESOL2Ø8Ø
                  DO 9 J=1,M                                           ESOL2Ø9Ø
                     NJ = N+J                                          ESOL21ØØ
  9                  AMOD(I,NJ) = B(1,I,J)                             ESOL211Ø
C                 OD                                                   ESOL212Ø
C              OD                                                      ESOL213Ø
 1Ø                                                           CONTINUEESOL214Ø
                                                              GOTO 17    ESOL215Ø
C           ELSE                                                         ESOL216Ø
 11            DO 16 I=1,N                                             ESOL217Ø
                  DO 13 J=1,N                                          ESOL218Ø
                     NTEMP = A(TT,I,J)                                 ESOL219Ø
                     DO 12 K=1,TT1                                     ESOL22ØØ
                        TK = TT - K                                    ESOL221Ø
 12                     NTEMP = MOD(PRIME(TK)*NTEMP+A(TK,I,J),P)        ESOL222Ø
C                    OD                                                ESOL223Ø
 13                  AMOD(I,J) = NTEMP                                 ESOL224Ø
C                 OD                                                   ESOL225Ø
                  DO 15 J=1,M                                          ESOL226Ø
                     NTEMP = B(TT,I,J)                                 ESOL227Ø
                     DO 14 K=1,TT1                                     ESOL228Ø
```

```
                    TK = TT - K                                          ESOL229Ø
 14                 NTEMP = MOD(PRIME(TK)*NTEMP+B(TK,I,J),P)             ESOL23ØØ
C                OD                                                      ESOL231Ø
                 NJ = N+J                                               ESOL232Ø
 15              AMOD(I,NJ) = NTEMP                                      ESOL233Ø
C             OD                                                        ESOL234Ø
C          OD                                                           ESOL235Ø
 16                                                                CONTINUEESOL236Ø
C                                                                  *     ESOL237Ø
C       END MAP ************************************************       ESOL238Ø
C                                                                        ESOL239Ø
C       PROCEDURE MSOLVE ***************************************       ESOL24ØØ
C          THIS PROCEDURE SOLVES THE SYSTEM OF EQUATIONS           *     ESOL241Ø
C          AX=B MODULO P BY COMPUTING                              *     ESOL242Ø
C             DMOD = DETERMINANT(A) MOD P                          *     ESOL243Ø
C          AND                                                     *     ESOL244Ø
C             A(ADJOINT)*B MOD P,                                  *     ESOL245Ø
C          WHICH IS TEMPORARILY STORED IN Y(ITER,I,J),             *     ESOL246Ø
C          I=1,2,...,N, J=1,2,...,M.                               *     ESOL247Ø
C                                                                  *     ESOL248Ø
C          THE GAUSSIAN ELIMINATION METHOD WITH PARTIAL            *     ESOL249Ø
C          PIVOTING IS USED TO REDUCE A TO UPPER ECHELON           *     ESOL25ØØ
C          FORM.                                                   *     ESOL251Ø
C                                                                  *     ESOL252Ø
C           NSING ASSUMES THE VALUE                                *     ESOL253Ø
C              Ø, IF A HAS RANK N, MOD P                           *     ESOL254Ø
C              1, IF A HAS RANK N-1, MOD P                         *     ESOL255Ø
C              2, IF A HAS RANK SMALLER THAN N-1, MOD P            *     ESOL256Ø
C                                                                  *     ESOL257Ø
C           KRAM IS EQUAL TO N IF A IS NONSINGULAR MODULO P;       *     ESOL258Ø
C              OTHERWISE, KRAM IS THE INDEX OF THE FIRST           *     ESOL259Ø
C              COLUMN WHICH CONTAINS A ZERO PIVOT ELEMENT.         *     ESOL26ØØ
C                                                                  *     ESOL261Ø
 17          DMOD = 1                                                    ESOL262Ø
             NSING = Ø                                                   ESOL263Ø
             II = Ø                                                      ESOL264Ø
C            REPEAT ... FORWARD ELIMINATION WITH II AS THE PIVOT         ESOL265Ø
C                  ... COLUMN                                            ESOL266Ø
 18             II = II+1                                               ESOL267Ø
                IF (NSING.LT.2 .AND. II.LE.N)                    GOTO 19 ESOL268Ø
                                                                 GOTO 34 ESOL269Ø
C               THEN                                                     ESOL27ØØ
 19                I = II - NSING                                        ESOL271Ø
C                  **ELIMINATION PROCEEDS ON THE SUBMATRIX WITH          ESOL272Ø
C                  **ROWS I,I+1,...,N AND COLUMNS II,II+1,...,N+M        ESOL273Ø
                   I1 = I+1                                              ESOL274Ø
                   II1 = II+1                                            ESOL275Ø
                   K = I-1                                               ESOL276Ø
C                  REPEAT ... SEARCH FOR NON-ZERO ELEMENT IN             ESOL277Ø
C                         ... II'TH COLUMN.                              ESOL278Ø
 2Ø                   K = K+1                                           ESOL279Ø
                      IF (K.LE.N)                                GOTO 21 ESOL28ØØ
                                                                 GOTO 22 ESOL281Ø
C                        THEN                                           ESOL282Ø
 21                        IF (AMOD(K,II).NE.Ø)                 GOTO 22 ESOL283Ø
                                                                 GOTO 2Ø ESOL284Ø
C                           THEN                                        ESOL285Ø
C                  ¢------------EXIT                                    ESOL286Ø
C                  CONTINUE                                              ESOL287Ø
 22                IF (K.LE.N)                                GOTO 23 ESOL288Ø
                                                              GOTO 33 ESOL289Ø
C                  THEN ... AMOD(K,II) IS THE PIVOT ELEMENT              ESOL29ØØ
 23                   IF (K.NE.I)                             GOTO 24 ESOL291Ø
                                                              GOTO 26 ESOL292Ø
C                     THEN ... INTERCHANGE ROWS I AND K                  ESOL293Ø
 24                      DO 25 J=II,NM                                   ESOL294Ø
                            NTEMP = AMOD(I,J)                            ESOL295Ø
                            AMOD(I,J) = AMOD(K,J)                        ESOL296Ø
 25                         AMOD(K,J) = NTEMP                           ESOL297Ø
C                       OD                                             ESOL298Ø
                        DMOD = -DMOD                                   ESOL299Ø
C                                                                      ESOL3ØØØ
C                     PROCEDURE INVERT ***************************     ESOL3Ø1Ø
C                        EUCLID'S EXTENDED ALGORITHM IS USED TO  *     ESOL3Ø2Ø
C                        FIND THE INVERSE, VI, MODULO P OF       *     ESOL3Ø3Ø
C                        AMOD(I,II).                             *     ESOL3Ø4Ø
 26                      V1 = 1                                         ESOL3Ø5Ø
```

```
                        U1 = Ø                                          ESOL3Ø6Ø
                        V3 = IABS(AMOD(I,II))                           ESOL3Ø7Ø
                        U3 = P                                          ESOL3Ø8Ø
C                       REPEAT                                          ESOL3Ø9Ø
 27                        IF (V3.NE.1)                          GOTO 28 ESOL31ØØ
                                                                GOTO 29 ESOL311Ø
C                         THEN                                          ESOL312Ø
 28                          Q = U3/V3                                   ESOL313Ø
                             T1 = U1 - Q*V1                              ESOL314Ø
                             T3 = U3 - Q*V3                              ESOL315Ø
                             U1 = V1                                     ESOL316Ø
                             U3 = V3                                     ESOL317Ø
                             V1 = T1                                     ESOL318Ø
                             V3 = T3                                     ESOL319Ø
                                                                GOTO 27 ESOL32ØØ
C                         CONTINUE                                      ESOL321Ø
 29                       IF (AMOD(I,II).LT.Ø) V1 = -V1                 ESOL322Ø
C                                                                *      ESOL323Ø
C                     END INVERT ********************************       ESOL324Ø
C                                                                       ESOL325Ø
                      DMOD = MOD(DMOD*AMOD(I,II),P)                     ESOL326Ø
                      DO 3Ø J=II1,NM                                    ESOL327Ø
 3Ø                      AMOD(I,J) = MOD(V1*AMOD(I,J),P)                ESOL328Ø
C                     OD                                                ESOL329Ø
                      IF (I.LT.N)                           GOTO 31 ESOL33ØØ
                                                            GOTO 34 ESOL331Ø
C                     THEN                                              ESOL332Ø
 31                      DO 32 K=I1,N                                   ESOL333Ø
                            NTEMP = AMOD(K,II)                          ESOL334Ø
                            DO 32 J=II1,NM                              ESOL335Ø
 32                            AMOD(K,J) = MOD(AMOD(K,J) - NTEMP*        ESOL336Ø
    1                                        AMOD(I,J),P)               ESOL337Ø
C                           OD                                          ESOL338Ø
C                       OD                                              ESOL339Ø
                                                            GOTO 18 ESOL34ØØ
C                   ELSE ... COLUMN II IS A ZERO PIVOT COLUMN.          ESOL341Ø
 33                    NSING = NSING + 1                                ESOL342Ø
                       IF (NSING.LT.2) KRAM = II                        ESOL343Ø
                                                            GOTO 18 ESOL344Ø
C             CONTINUE                                                  ESOL345Ø
 34           IF (NSING.LT.2)                               GOTO 35 ESOL346Ø
                                                            GOTO 42 ESOL347Ø
C             THEN ... RANK OF A MODULO P IS N OR N-1; THEREFORE        ESOL348Ø
C                  ... BACK SUBSTITUTE.                                 ESOL349Ø
 35              IF (NSING.EQ.Ø) KRAM = N                               ESOL35ØØ
                 KRAM1 = KRAM - 1                                       ESOL351Ø
                 KRAM2 = KRAM + 1                                       ESOL352Ø
                 DO 41 J=N1,NM                                          ESOL353Ø
                    NJ = J - N                                          ESOL354Ø
                    IF (KRAM.NE.N)                          GOTO 36 ESOL355Ø
                                                            GOTO 38 ESOL356Ø
C                   THEN                                                ESOL357Ø
 36                    DO 37 I=KRAM2,N                                  ESOL358Ø
 37                       Y(ITER,I,NJ) = Ø                              ESOL359Ø
C                     OD                                                ESOL36ØØ
 38                  NTEMP = MOD(DMOD*AMOD(N,J),P)                     ESOL361Ø
                     IF (NSING.EQ.1) NTEMP = NTEMP*(-1)**(N-KRAM)      ESOL362Ø
                     Y(ITER,KRAM,NJ) = NTEMP                           ESOL363Ø
                     DO 4Ø II=1,KRAM1                                  ESOL364Ø
                        I = KRAM - II                                  ESOL365Ø
                        I1 = I+1                                       ESOL366Ø
                        NTEMP = Ø                                      ESOL367Ø
                        DO 39 K=I1,KRAM                                ESOL368Ø
 39                        NTEMP = MOD(NTEMP+AMOD(I,K)*                ESOL369Ø
    1                                Y(ITER,K,NJ), P)                  ESOL37ØØ
C                       OD                                             ESOL371Ø
                        NTEMP = -NTEMP                                 ESOL372Ø
                        IF (NSING.EQ.Ø) NTEMP = MOD(NTEMP + DMOD*      ESOL373Ø
    1                                           AMOD(I,J),P)           ESOL374Ø
 4Ø                     Y(ITER,I,NJ) = NTEMP                           ESOL375Ø
C                    OD                                                ESOL376Ø
C                 OD                                                   ESOL377Ø
 41                                                         CONTINUEESOL378Ø
                                                            GOTO 44 ESOL379Ø
C             ELSE ... RANK OF A MODULO P IS LESS THAN N-1.             ESOL38ØØ
 42              DO 43 J=1,M                                            ESOL381Ø
```

```
                DO 43 I=1,N                                                ESOL382Ø
 43                Y(ITER,I,J) = Ø                                         ESOL383Ø
C                  OD                                                      ESOL384Ø
C               OD                                                         ESOL385Ø
 44           IF (NSING.NE.Ø) DMOD = Ø                                     ESOL386Ø
C                                                                  *       ESOL387Ø
C       END MSOLVE ***************************************************     ESOL388Ø
C                                                                          ESOL389Ø
C       PROCEDURE MIXED-RADIX ********************************************ESOL39ØØ
C          THIS PROCEDURE COMPUTES THE ITER'TH TERMS OF THE        *       ESOL391Ø
C          MIXED-RADIX REPRESENTATION OF Y( ,N,M) AND DET          *       ESOL392Ø
C          USING THE CHINESE REMAINDER THEOREM.                    *       ESOL393Ø
C                                                                  *       ESOL394Ø
C           TEST IS TRUE ONLY IF THE ITER'TH TERMS OF THE          *       ESOL395Ø
C           MIXED-RADIX REPRESENTATIONS OF DET AND Y ARE           *       ESOL396Ø
C           ALL ZERO.                                              *       ESOL397Ø
C                                                                  *       ESOL398Ø
C           NOTE: IN FORTRAN, THE EVALUATION B=MOD(A,P) YEILDS     *       ESOL399Ø
C           AN INTEGER B SUCH THAT -P¢B¢P. IN THIS PROCEDURE,      *       ESOL4ØØØ
C           ANY SUCH B IS CONVERTED TO A SYMMETRIC RESIDUE C,      *       ESOL4Ø1Ø
C           -P¢2C¢P VIA THE EVALUATION C=B-(B/P2)*P.               *       ESOL4Ø2Ø
C                                                                  *       ESOL4Ø3Ø
            TEST = .TRUE.                                                  ESOL4Ø4Ø
            ITER1 = ITER - 1                                               ESOL4Ø5Ø
            ITER2 = ITER - 2                                               ESOL4Ø6Ø
            DO 52 I=1,N                                                    ESOL4Ø7Ø
               DO 52 J=1,M                                                 ESOL4Ø8Ø
                  IF (ITER.EQ.1)                                 GOTO 45 ESOL4Ø9Ø
                                                                 GOTO 46 ESOL41ØØ
C                 THEN                                                     ESOL411Ø
 45                  MULT = Y(ITER,I,J)                                    ESOL412Ø
                                                                 GOTO 5Ø ESOL413Ø
C                 ELSE                                                     ESOL414Ø
 46                  MULT = Y(ITER1,I,J)                                   ESOL415Ø
                     IF (ITER.NE.2)                              GOTO 47 ESOL416Ø
                                                                 GOTO 49 ESOL417Ø
C                    THEN                                                  ESOL418Ø
 47                     DO 48 LL=1,ITER2                                   ESOL419Ø
                           L = ITER1 - LL                                  ESOL42ØØ
 48                        MULT = MOD(MULT*PRIME(L)+Y(L,I,J),P)            ESOL421Ø
C                       OD                                                 ESOL422Ø
 49                  MULT = MOD(IP * MOD(Y(ITER,I,J)-MULT,P),P)            ESOL423Ø
 5Ø               Y(ITER,I,J) = MULT - (MULT/P2)*P                         ESOL424Ø
                  IF (TEST)                                      GOTO 51 ESOL425Ø
                                                                 GOTO 52 ESOL426Ø
C                 THEN                                                     ESOL427Ø
 51                  IF (Y(ITER,I,J).NE.Ø) TEST = .FALSE.                  ESOL428Ø
 52                                                              CONTINUEESOL429Ø
C              OD                                                          ESOL43ØØ
C           OD                                                             ESOL431Ø
            IF (ITER.EQ.1)                                     GOTO 53 ESOL432Ø
                                                               GOTO 54 ESOL433Ø
C           THEN                                                           ESOL434Ø
 53            MULT = DMOD                                                 ESOL435Ø
                                                               GOTO 58 ESOL436Ø
C           ELSE                                                           ESOL437Ø
 54            MULT = DET(ITER1)                                           ESOL438Ø
               IF (ITER.NE.2)                                  GOTO 55 ESOL439Ø
                                                               GOTO 57 ESOL44ØØ
C              THEN                                                        ESOL441Ø
 55               DO 56 LL=1,ITER2                                         ESOL442Ø
                     L = ITER1 - LL                                        ESOL443Ø
 56                  MULT = MOD(MULT*PRIME(L) + DET(L),P)                  ESOL444Ø
C                 OD                                                       ESOL445Ø
 57            MULT = MOD(IP * MOD(DMOD-MULT,P), P)                        ESOL446Ø
 58         DET(ITER) = MULT - (MULT/P2)*P                                 ESOL447Ø
            IF (DET(ITER).NE.Ø) TEST = .FALSE.                             ESOL448Ø
C                                                                  *       ESOL449Ø
C       END MIXED-RADIX **************************************************ESOL45ØØ
C                                                                          ESOL451Ø
         IF (TEST)                                             GOTO 59 ESOL452Ø
                                                               GOTO 6Ø ESOL453Ø
C        THEN                                                            ESOL454Ø
 59         NOZERO = NOZERO + 1                                          ESOL455Ø
                                                               GOTO 61 ESOL456Ø
C        ELSE                                                            ESOL457Ø
```

```
 6Ø         NOZERO = Ø                                                    ESOL458Ø
 61      IF (RTEST .AND. ITER.GT.NOZERO .AND. NOZERO.GE.TR)      GOTO 62 ESOL459Ø
                                                                 GOTO 64 ESOL46ØØ
C        THEN ... NORMAL EXIT; FIRST, HOWEVER CHECK FOR ZERO             ESOL461Ø
C             ... DETERMINANT.                                           ESOL462Ø
 62         IER = Ø                                                      ESOL463Ø
            NOCOEF = ITER - NOZERO                                       ESOL464Ø
            DO 63 K=1,NOCOEF                                             ESOL465Ø
               IF (DET(K).NE.Ø) RETURN                                   ESOL466Ø
C           OD                                                           ESOL467Ø
 63                                                             CONTINUEESOL468Ø
            IER = 2                                                      ESOL469Ø
            RETURN                                                       ESOL47ØØ
 64   CONTINUE                                                           ESOL471Ø
C                                                                        ESOL472Ø
C     **MAXPRM COEFFICIENTS HAVE BEEN USED, ADNORMAL EXIT.               ESOL473Ø
C                                                                        ESOL474Ø
      NOCOEF = MAXPRM - NOZERO                                           ESOL475Ø
      IER = 1                                                            ESOL476Ø
      IF (NOCOEF.EQ.Ø) IER = 3                                           ESOL477Ø
      RETURN                                                             ESOL478Ø
      END                                                                ESOL479Ø


      BLOCK DATA                                                         DATAØØ1Ø
C                                                                        DATAØØ2Ø
C*****************************************************************     DATAØØ3Ø
C                                                                *       DATAØØ4Ø
C                  *****WARNING*****                            *       DATAØØ5Ø
C                                                               *       DATAØØ6Ø
C ARRAY NAMES ARE USED IN THE DATA STATEMENTS BELOW TO          *       DATAØØ7Ø
C SPECIFY VALUES FOR THE ARRAYS KPRIME AND IPRIME. IN SOME      *       DATAØØ8Ø
C INSTALLATION IT MAY BE NECESSARY TO EXPLICITLY LIST THE       *       DATAØØ9Ø
C COMPONENTS OF THE ARRAYS, NAMELY, KPRIME(1), KPRIME(2),...,   *       DATAØ1ØØ
C KPRIME(1ØØ) AND IPRIME(1), IPRIME(2),...,IPRIME(1ØØ).         *       DATAØ11Ø
C                                                               *       DATAØ12Ø
C*****************************************************************     DATAØ13Ø
C                                                                        DATAØ14Ø
      COMMON /PRIMEB/KPRIME(1ØØ),IPRIME(1ØØ)                             DATAØ15Ø
      DATA KPRIME/       45233,45247,45259,45263,45281,45289             DATAØ16Ø
     1,45293,453Ø7,45317,45319,45329,45337,45341,45343,45361             DATAØ17Ø
     2,45377,45389,454Ø3,45413,45427,45433,45439,45481,45491             DATAØ18Ø
     3,45497,455Ø3,45523,45533,45541,45553,45557,45569,45587             DATAØ19Ø
     4,45589,45599,45613,45631,45641,45659,45667,45673,45677             DATAØ2ØØ
     5,45691,45697,457Ø7,45737,45751,45757,45763,45767,45779             DATAØ21Ø
     6,45817,45821,45823,45827,45833,45841,45853,45863,45869             DATAØ22Ø
     7,45887,45893,45943,45949,45953,45959,45971,45979,45989             DATAØ23Ø
     8,46Ø21,46Ø27,46Ø49,46Ø51,46Ø61,46Ø73,46Ø91,46Ø93,46Ø99             DATAØ24Ø
     9,461Ø3,46133,46141,46147,46153,46171,46181,46183,46187             DATAØ25Ø
     X,46199,46219,46229,46237,46261,46271,46273,46279,463Ø1             DATAØ26Ø
     Y,463Ø7,463Ø9,46327,46337/                                          DATAØ27Ø
      DATA IPRIME/       ØØØØØ,42Ø15,28577,Ø11Ø8,29342,16641             DATAØ28Ø
     1,1Ø4Ø5,19447,26685,39525,14116,12753,32178,Ø1Ø43,Ø8857             DATAØ29Ø
     2,27911,15Ø49,Ø7Ø79,33425,ØØ8Ø4,23175,23886,44779,41942             DATAØ3ØØ
     3,1Ø171,166Ø6,1Ø638,17371,27195,35827,42639,Ø1829,24658             DATAØ31Ø
     4,Ø9Ø23,37958,3Ø638,Ø6339,4127Ø,4Ø538,1Ø157,11783,ØØ457             DATAØ32Ø
     5,32947,4217Ø,1791Ø,33474,2ØØ17,25Ø86,365Ø8,37444,35543             DATAØ33Ø
     6,Ø6993,1Ø326,16328,26765,42Ø83,37223,3Ø711,Ø94Ø8,Ø6635             DATAØ34Ø
     7,38421,11397,32683,17333,34245,15748,35735,23492,193Ø2             DATAØ35Ø
     8,2ØØ76,4562Ø,44978,Ø9864,14832,16Ø92,19457,24Ø45,4495Ø             DATAØ36Ø
     9,32872,243Ø9,15726,43Ø57,37766,14Ø46,41826,19946,41363             DATAØ37Ø
     X,23967,39791,29237,18Ø85,12952,3685Ø,Ø2213,3ØØ23,34871             DATAØ38Ø
     Y,42667,4Ø41Ø,32615,46136/                                          DATAØ39Ø
      END                                                                DATAØ4ØØ



      SUBROUTINE SUBBND(A,B,GAMMA,DELTA,S,SUM,T,N,M,NM,                  SBNDØØ1Ø
     1                  TWOT1,BOUND,NO)                                  SBNDØØ2Ø
C                                                                        SBNDØØ3Ø
C*****************************************************************     SBNDØØ4Ø
C                                                               *        SBNDØØ5Ø
C USING HADAMARD'S INEQUALITY, THIS SUBROUTINE COMPUTES THE     *       SBNDØØ6Ø
C MAXIMUM NUMBER OF PRIMES REQUIRED TO REPRESENT DETERMINANT(A)*        SBNDØØ7Ø
C AND A(ADJOINT)*B FOR THE SYSTEM OF LINEAR EQUATIONS AX=B,     *       SBNDØØ8Ø
C WHERE A( ,N,N) AND B( ,N,M) ARE MATRICES WITH MULTIPLE-       *       SBNDØØ9Ø
```

```
C PRECISION INTEGER COEFFICIENTS EXPRESSED IN MIXED-RADIX FORM.*        SBND01ØØ
C                                                              *        SBND0110
C  PRIME  IS A LINEAR ARRAY CONTAINING 1ØØ DISTINCT PRIME      *        SBND0120
C (INPUT) INTEGERS IN ASCENDING ORDER.  THESE PRIMES ARE THE   *        SBND0130
C         RADII USED IN THE REPRESENTATION OF A AND B AND OF   *        SBND0140
C         DETERMINANT(A) AND A(ADJOINT)*B.THE PRIMES ARE CHOSEN*        SBND0150
C         AS LARGE AS POSSIBLE SUBJECT TO THE CONDITION THAT   *        SBND0160
C         FOR ALL I AND J PRIME(I)*PRIME(J) DOES NOT OVERFLOW  *        SBND0170
C         AN INTEGER WORD.                                     *        SBND0180
C      A  IS AN INTEGER MATRIX OF DIMENSION T BY N BY N.  THE  *        SBND0190
C (INPUT) FIRST DIMENSION CONTAINS THE COEFFICIENTS OF THE     *        SBND0200
C         MIXED-RADIX REPRESENTATION OF THE MULTIPLE-PRECISION *        SBND0210
C         COMPONENTS OF THE N BY N MATRIX A( ,N,N).  THAT IS,  *        SBND0220
C            A( ,I,J) = A(1,I,J)                               *        SBND0230
C                     + A(2,I,J)*PRIME(1)                      *        SBND0240
C                     .                                        *        SBND0250
C                     .                                        *        SBND0260
C                     .                                        *        SBND0270
C                     + A(T,I,J)*PRIME(1)*...*PRIME(T-1),      *        SBND0280
C         FOR I,J=1,2,...,N.                                   *        SBND0290
C      B  IS AN INTEGER MATRIX OF DIMENSION T BY N BY M WITH   *        SBND0300
C (INPUT) A SIMILAR NOTATIONAL CORRESPONDENCE MADE FOR A ABOVE.*        SBND0310
C  DELTA  IS AN INTEGER ARRAY OF DIMENSION TWOT1 USED TO STORE *        SBND0320
C  (TEMP) THE COEFFICIENTS OF THE MULTIPLE-PRECISION INTEGERS  *        SBND0330
C         A( ,I,J)**2 AND B( ,I,J)**2 FOR ANY PARTICULAR       *        SBND0340
C         A( ,I,J) OR B( ,I,J).                                *        SBND0350
C  GAMMA  IS AN INTEGER ARRAY OF DIMENSION TWOT1 USED TO STORE *        SBND0360
C  (TEMP) THE COEFFICIENTS OF THE MULTIPLE-PRECISION PARTIAL   *        SBND0370
C         PRODUCTS OBTAINED WHEN A( ,I,J)**2 OR B( ,I,J)**2 ARE*        SBND0380
C         BEING COMPUTED.                                      *        SBND0390
C    SUM  IS AN INTEGER ARRAY OF DIMENSION TWOT1 USED TO STORE *        SBND0400
C  (TEMP) THE SUM OF (A( ,I,J)**2, I=1,...,N) OR THE SUM OF    *        SBND0410
C         (B( ,I,J)**2 ,I=1,...,N) FOR ANY PARTICULAR COLUMN J.*        SBND0420
C      S  IS A REAL ARRAY OF DIMENSION NM. FOR 1<=J<=N, S(J)   *        SBND0430
C  (TEMP) CONTAINS A BOUND FOR THE LOGARITHM OF SUM(A( ,I,J)**2*        SBND0440
C         ,I=1,...,N) AND FOR N+1<=J<=NM OF SUM(B( ,I,J)**2,   *        SBND0450
C         I=1,...,N).                                          *        SBND0460
C      T  IS THE NUMBER OF RADII, PRIME(1),...,PRIME(T), USED  *        SBND0470
C (INPUT) TO REPRESENT EACH COMPONENT OF A( ,N,N) AND B( ,N,M) *        SBND0480
C         IN MIXED-RADIX FORM.                                 *        SBND0490
C      N  IS THE NUMBER OF EQUATIONS AND THE NUMBER OF UNKNOWNS*        SBND0500
C (INPUT) IN THE SYSTEM.  (I.E., N IS THE SIZE OF THE SECOND   *        SBND0510
C         AND THIRD DIMENSIONS OF A.)                          *        SBND0520
C      M  IS THE NUMBER OF RIGHT-HAND VECTORS FOR WHICH THE    *        SBND0530
C (INPUT) SYSTEM IS TO BE SOLVED.  (I.E., M IS THE SIZE OF THE *        SBND0540
C         THIRD DIMENSION OF B.)                               *        SBND0550
C     NM  IS EQUAL TO N+M.                                     *        SBND0560
C (INPUT)                                                      *        SBND0570
C  TWOT1  IS EQUAL TO 2*T+1.                                   *        SBND0580
C (INPUT)                                                      *        SBND0590
C    NO   IS A BOUND FOR THE NUMBER OF PRIMES REQUIRED TO SOLVE*        SBND0600
C(OUTPUT) THE SYSTEM OF LINEAR EQUATIONS AX=B. THAT IS, NO IS  *        SBND0610
C         A BOUND FOR THE NUMBER OF PRIMES REQUIRED TO         *        SBND0620
C         REPRESENT DETERMINANT(A) AND A(ADJOINT)*B IN MIXED-  *        SBND0630
C         RADIX FORM.                                          *        SBND0640
C   BOUND IS EQUAL TO THE LOGARITHM OF THE PRODUCT OF THE FIRST*        SBND0650
C(OUTPUT) NO PRIMES.                                           *        SBND0660
C                                                              *        SBND0670
C***************************************************************        SBND0680
C                                                                       SBND0690
      INTEGER T,TWOT,TWOT1,A(T,N,N),B(T,N,M),DELTA(TWOT1),              SBND0700
     1        GAMMA(TWOT1),PRIME(1ØØ),SUM(TWOT1),CSUM,Q,PD              SBND0710
      REAL S(NM)                                                        SBND0720
      INTEGER  TK,TK1                                                   
      COMMON /PRIMEB/ PRIME, IPRIME(1ØØ)                                SBND 730
      TWOT = 2*T                                                        SBND0740
C     PROCEDURE ABOUND*********************************************     SBND0750
C        FOR EACH J, 1¢=J¢=N, THE SUM OF THE SQUARES OF THE     *       SBND0760
C        ELEMENTS IN THE  J-TH COLUMN OF A IS COMPUTED. THE     *       SBND0770
C        SUM, WHICH IS A MIXED-RADIX, MULTIPLE-PRECISION        *       SBND0780
C        INTEGER OF LENGTH AT MOST 2*T+1, IS STORED IN THE      *       SBND0790
C        VECTOR SUM. THE LOGARITHM OF A BOUND FOR THE SUM IS    *       SBND0800
C        THEN COMPUTED AND STORED IN S(J).                      *       SBND0810
C                                                               *       SBND0820
         DO 16 J=1,N                                                    SBND0830
```

```
          **COMPUTE LOG (BOUND FOR (SUM OF A( ,I,J)**2,                SBND084Ø
C         **I=1,...,N)), AND STORE THE RESULTS IN S(J).                SBND085Ø
          DO 1  K=1,TWOT1                                              SBND086Ø
 1           SUM(K) = Ø                                                SBND087Ø
C         OD                                                           SBND088Ø
          DO 9 I=1,N                                                   SBND089Ø
C            PROCEDURE SQUARE*********************************        SBND09ØØ
C               A( ,I,J)**2 IS COMPUTED. THE COEFFICIENTS OF *        SBND091Ø
C               THE MIXED-RADIX, MULTIPLE-PRECISION RESULT   *        SBND092Ø
C               ARE STORED IN DELTA.                         *        SBND093Ø
C                                                            *        SBND094Ø
                DO 2  II=1,TWOT1                                       SBND095Ø
                   DELTA(II) = Ø                                       SBND096Ø
 2                 GAMMA(II) = Ø                                       SBND097Ø
C               OD                                                     SBND098Ø
                Q = Ø                                                  SBND099Ø
                K = Ø                                                  SBND1ØØØ
                L = T-1                                                SBND1Ø1Ø
C               REPEAT                                                 SBND1Ø2Ø
 3                 IF (K.LE.L)                                 GOTO 4   SBND1Ø3Ø
                                                               GOTO 7   SBND1Ø4Ø
C                 THEN                                                 SBND1Ø5Ø
C                    **COMPUTE PARTIAL PRODUCT AND STORE IN            SBND1Ø6Ø
C                    **GAMMA.                                          SBND1Ø7Ø
 4                   Q = Ø                                             SBND1Ø8Ø
                     TK = T-K                                          SBND1Ø9Ø
                     DO 5  II=1,T                                      SBND11ØØ
                        PD = A(TK,I,J)*A(II,I,J) + Q                   SBND111Ø
                        Q = PD/PRIME(II)                               SBND112Ø
 5                      GAMMA(II) = PD - Q*PRIME(II)                  SBND113Ø
C                    OD                                                SBND114Ø
                     GAMMA(T+1) = Q                                    SBND115Ø
C                    **ADD PARTIAL PRODUCT TO DELTA.                   SBND116Ø
                     Q = Ø                                             SBND117Ø
                     L1 = T+K                                          SBND118Ø
                     TK1 = T+K+1                                       SBND119Ø
                     DO 6  II=1,L1                                     SBND12ØØ
                        PD = PRIME(TK)*DELTA(II) + GAMMA(II)           SBND121Ø
     1                       + Q                                       SBND122Ø
                        Q = PD/PRIME(II)                               SBND123Ø
 6                      DELTA(II) = PD - Q*PRIME(II)                  SBND124Ø
C                    OD                                                SBND125Ø
                     DELTA(TK1) = Q                                    SBND126Ø
                     K = K+1                                           SBND127Ø
                                                               GOTO 3   SBND128Ø
C                 CONTINUE                                             SBND129Ø
C                                                            *        SBND13ØØ
C            END SQUARE***************************************        SBND131Ø
C                                                                     SBND132Ø
C                                                                     SBND133Ø
C            **ACCUMULATE THE SUM OF A( ,I,J)**2.                     SBND134Ø
 7           Q = Ø                                                    SBND135Ø
             DO 8 II=1,TWOT1                                          SBND136Ø
                CSUM = DELTA(II)+SUM(II)+Q                           SBND137Ø
                Q = CSUM/PRIME(II)                                   SBND138Ø
 8              SUM(II) = CSUM - Q*PRIME(II)                         SBND139Ø
C            OD                                                       SBND14ØØ
C         OD                                                          SBND141Ø
 9                                                           CONTINUESBND142Ø
C                                                                     SBND143Ø
C                                                                     SBND144Ø
C         **CALCULATE ABOUND FOR THE MULTIPLE-PRECISION               SBND145Ø
C         **INTEGER IN SUM AND STORE THE LOG OF THIS                  SBND146Ø
C         **BOUND IN S(J).                                            SBND147Ø
          K = TWOT1                                                   SBND148Ø
C         REPEAT...FIND OUT THE LENGTH K OF THE MULTIPLE-             SBND149Ø
C               ...PRECISION NUMBER IN SUM.                           SBND15ØØ
 1Ø          IF (K.GT.1.AND.SUM(K).EQ.Ø)                  GOTO 11 SBND151Ø
                                                          GOTO 12 SBND152Ø
C            THEN                                                     SBND153Ø
 11             K = K - 1                                            SBND154Ø
                                                          GOTO 1Ø SBND155Ø
C         CONTINUE                                                    SBND156Ø
 12       S(J) = Ø.Ø                                                  SBND157Ø
          IF (K.GT.1)                                  GOTO 13 SBND158Ø
                                                       GOTO 15 SBND159Ø
```

```
C           THEN                                                        SBND16ØØ
 13            L = K-1                                                  SBND161Ø
               DO 14 II=1,L                                             SBND162Ø
 14               S(J) = S(J) +ALOG(FLOAT(PRIME(II)))                   SBND163Ø
C              OD                                                       SBND164Ø
               S(J) = S(J)+ALOG(FLOAT(SUM(K)+1))                        SBND165Ø
                                                                  GOTO 16 SBND166Ø
C           ELSE                                                        SBND167Ø
 15            IF (SUM(K).GT.Ø) S(J) = ALOG(FLOAT(SUM(K)))              SBND168Ø
C        OD                                                             SBND169Ø
 16                                                            CONTINUESBND17ØØ
C                                                                *      SBND171Ø
C     END ABOUND*************************************************       SBND172Ø
C                                                                        SBND173Ø
C                                                                        SBND174Ø
C     PROCEDURE BBOUND*******************************************       SBND175Ø
C        FOR EACH J, 1ç=Jç=M, THE SUM OF THE SQUARES OF THE      *       SBND176Ø
C        ELEMENTS IN THE  J-TH COLUMN OF B IS COMPUTED. THE      *       SBND177Ø
C        SUM, WHICH IS A MIXED-RADIX, MULTIPLE PRECISION         *       SBND178Ø
C        INTEGER OF LENGTH AT MOST 2*T+1, IS STORED IN THE       *       SBND179Ø
C        VECTOR SUM. THE LOGARITHM OF A BOUND FOR THE SUM IS     *       SBND18ØØ
C        THEN COMPUTED AND STORED IN S(N+J).                     *       SBND181Ø
C                                                                *       SBND182Ø
C        NOTE: THE COMPUTATIONAL PROCEDURE IS THE SAME AS FOR    *       SBND183Ø
C              A ABOVE.                                          *       SBND184Ø
C                                                                *       SBND185Ø
         DO 32 J=1,M                                                    SBND186Ø
            DO 17 K=1,TWOT1                                             SBND187Ø
 17            SUM(K) = Ø                                               SBND188Ø
C           OD                                                          SBND189Ø
            DO 25 I=1,N                                                 SBND19ØØ
               DO 18 II=1,TWOT1                                         SBND191Ø
                  DELTA(II) = Ø                                         SBND192Ø
 18               GAMMA(II) = Ø                                         SBND193Ø
C              OD                                                       SBND194Ø
               Q = Ø                                                    SBND195Ø
               K = Ø                                                    SBND196Ø
               L = T-1                                                  SBND197Ø
C              REPEAT                                                   SBND198Ø
 19               IF (K.LE.L)                                          GOTO 2Ø SBND199Ø
                                                                       GOTO 23 SBND2ØØØ
C                  THEN                                                 SBND2Ø1Ø
 2Ø                   Q = Ø                                            SBND2Ø2Ø
                      TK = T-K                                         SBND2Ø3Ø
                      DO 21 II=1,T                                     SBND2Ø4Ø
                         PD = B(TK,I,J)*B(II,I,J) + Q                  SBND2Ø5Ø
                         Q = PD/PRIME(II)                              SBND2Ø6Ø
 21                      GAMMA(II) = PD - Q*PRIME(II)                  SBND2Ø7Ø
C                     OD                                               SBND2Ø8Ø
                      GAMMA(T+1) = Q                                   SBND2Ø9Ø
                      Q = Ø                                            SBND21ØØ
                      L1 = T+K                                         SBND211Ø
                      TK1 = T+K+1                                      SBND212Ø
                      DO 22 II=1,L1                                    SBND213Ø
                         PD = PRIME(TK)*DELTA(II) + GAMMA(II)+Q        SBND214Ø
                         Q = PD/PRIME(II)                              SBND215Ø
 22                      DELTA(II) = PD - Q*PRIME(II)                  SBND216Ø
C                     OD                                               SBND217Ø
                      DELTA(TK1) = Q                                   SBND218Ø
                      K = K+1                                          SBND219Ø
                                                                  GOTO 19 SBND22ØØ
C              CONTINUE                                                 SBND221Ø
 23            Q = Ø                                                    SBND222Ø
               DO 24 II=1,TWOT1                                         SBND223Ø
                  CSUM = DELTA(II)+SUM(II)+Q                            SBND224Ø
                  Q = CSUM/PRIME(II)                                    SBND225Ø
 24               SUM(II) = CSUM - Q*PRIME(II)                          SBND226Ø
C              OD                                                       SBND227Ø
C           OD                                                          SBND228Ø
 25                                                            CONTINUESBND229Ø
            K = TWOT1                                                   SBND23ØØ
C           REPEAT                                                      SBND231Ø
 26            IF (K.GT.1.AND.SUM(K).EQ.Ø)                             GOTO 27 SBND232Ø
                                                                       GOTO 28 SBND233Ø
C                THEN                                                   SBND234Ø
 27                K = K - 1                                           SBND235Ø
```

```
                                                                  GOTO 26 SBND236Ø
C           CONTINUE                                                      SBND237Ø
 28         NJ = N+J                                                      SBND238Ø
            S(NJ) = Ø.Ø                                                   SBND239Ø
            IF (K.GT.1)                                           GOTO 29 SBND24ØØ
                                                                  GOTO 31 SBND241Ø
C           THEN                                                          SBND242Ø
 29            L = K-1                                                   SBND243Ø
               DO 3Ø II=1,L                                             SBND244Ø
 3Ø               S(NJ) = S(NJ) +ALOG(FLOAT(PRIME(II)))                 SBND245Ø
C              OD                                                        SBND246Ø
               S(NJ) = S(NJ)+ALOG(FLOAT(SUM(K)+1))                      SBND247Ø
                                                                  GOTO 32 SBND248Ø
C           ELSE                                                          SBND249Ø
 31            IF (SUM(K).GT.Ø) S(J) = ALOG(FLOAT(SUM(K)))               SBND25ØØ
C        OD                                                               SBND251Ø
 32                                                               CONTINUESBND252Ø
C                                                                *        SBND253Ø
C     END BBOUND**************************************************        SBND254Ø
C                                                                         SBND255Ø
C                                                                         SBND256Ø
C     **FIND MIN(S(I)), I=1,...,N.                                        SBND257Ø
         SMIN = S(1)                                                      SBND258Ø
         DO 33 I=1,N                                                      SBND259Ø
 33         IF (SMIN.GT.S(I)) SMIN = S(I)                                SBND26ØØ
C        OD                                                               SBND261Ø
C     **FIND MAX(S(I)), I=N+1,...,NM.                                     SBND262Ø
         SMAX = S(N+1)                                                    SBND263Ø
         L = N+1                                                          SBND264Ø
         DO 34 I=L,NM                                                     SBND265Ø
 34         IF (SMAX.LT.S(I)) SMAX = S(I)                                SBND266Ø
C        OD                                                               SBND267Ø
C     **IF SMIN ¼= SMAX, THEN BOUND = (SUM OF S(I))/2+ LOG(2.Ø)           SBND268Ø
C     **ELSE BOUND = ((SUM OF S(I))+SMAX-SMIN)/2+LOG(2.Ø),                SBND269Ø
C     **WHERE THE SUM IS TAKEN OVER I=1,...,N.                            SBND27ØØ
         BOUND = Ø.Ø                                                      SBND271Ø
         DO 35 I=1,N                                                      SBND272Ø
 35         BOUND = BOUND+S(I)                                           SBND273Ø
C        OD                                                               SBND274Ø
         IF (SMIN.GE.SMAX)                                       GOTO 36 SBND275Ø
                                                                 GOTO 37 SBND276Ø
C        THEN                                                             SBND277Ø
 36         BOUND = BOUND/2.Ø + ALOG(2.Ø)                                SBND278Ø
                                                                 GOTO 38 SBND279Ø
C        ELSE                                                             SBND28ØØ
 37         BOUND = (BOUND+SMAX-SMIN)/2.Ø + ALOG(2.Ø)                    SBND281Ø
C     **CALCULATE NUMBER OF PRIMES REQUIRED.                              SBND282Ø
 38      NO = Ø                                                           SBND283Ø
         SUMLOG = Ø.Ø                                                     SBND284Ø
C        REPEAT                                                           SBND285Ø
 39         NO = NO+1                                                    SBND286Ø
            IF (SUMLOG.GE.BOUND)                                 GOTO 4Ø SBND287Ø
C           THEN                                                         SBND288Ø
C        ¢------EXIT                                                     SBND289Ø
            SUMLOG = SUMLOG + ALOG(FLOAT(PRIME(NO)))                     SBND29ØØ
                                                                 GOTO 39 SBND291Ø
C        CONTINUE                                                         SBND292Ø
 4Ø   RETURN                                                             SBND293Ø
      END                                                                SBND294Ø

      SUBROUTINE MRADIX(A,N,C,LMAX,L,B,IER)                              MRADØØ1Ø
C                                                                        MRADØØ2Ø
C*****************************************************************      MRADØØ3Ø
C GIVEN THE FIXED-RADIX INTEGER                                  *       MRADØØ4Ø
C                                                                *       MRADØØ5Ø
C   A(1)*B**(N-1) + ... + A(N-1)*B + A(N),                       *       MRADØØ6Ø
C                                                                *       MRADØØ7Ø
C WHERE ABS(A(I)) < B, I=1,...,N,                                *       MRADØØ8Ø
C THIS SUBROUTINE COMPUTES(FOR SOME L <= LMAX) THE COEFFICIENTS  *       MRADØØ9Ø
C C(1),C(2),...,C(L) IN ITS SYMMETRIC MIXED-RADIX                *       MRADØ1ØØ
C REPRESENTATION:                                                *       MRADØ11Ø
C                                                                *       MRADØ12Ø
C   C(1)+C(2)*P(1) + ... + C(L)*P(1)*P(2)*...*P(L-1).            *       MRADØ13Ø
C                                                                *       MRADØ14Ø
C LMAX MUST BE GIVEN SUCH THAT                                   *       MRADØ15Ø
C                                                                *       MRADØ16Ø
```

```
C   LMAX >= (N * LOG B + LOG 2) / LOG(P(1)).                    *       MRADØ17Ø
C                                                               *       MRADØ18Ø
C IER IS AN ERROR CODE WHICH IS 1 IF THE DIMENSION PARAMETER N,*       MRADØ19Ø
C LMAX ARE INCORRECT, AND Ø OTHERWISE.                          *       MRADØ2ØØ
C                                                               *       MRADØ21Ø
C                    ******* WARNING  *******                   *       MRADØ22Ø
C                                                               *       MRADØ23Ø
C THIS SUBROUTINE ASSUMMES THAT FOR ALL I,J                     *       MRADØ24Ø
C                                                               *       MRADØ25Ø
C   P(I)*P(J) AND P(I)*B                                        *       MRADØ26Ø
C                                                               *       MRADØ27Ø
C DO NOT OVERFLOW A COMPUTER WORD.                              *       MRADØ28Ø
C                                                               *       MRADØ29Ø
C***************************************************************       MRADØ3ØØ
C                                                                       MRADØ31Ø
      INTEGER A(N),C(LMAX),P(1ØØ),Q,QTEMP,PP,B,P1                       MRADØ32Ø
      COMMON /PRIMEB/ P, IPRIME(1ØØ)                                    MRAD 33Ø
      IER = 1                                                           MRADØ34Ø
      IF (N.LT.1 .OR. LMAX.LT.1) RETURN                                 MRADØ35Ø
      L = 1                                                             MRADØ36Ø
      C(1) = Ø                                                          MRADØ37Ø
      DO 4 I=1,N                                                        MRADØ38Ø
C        **AT THIS STAGE C(1)+C(2)*P(1)+...+C(L)*P(1)*P(2)*...*         MRADØ39Ø
C        **P(L-1) IS THE MIXED-RADIX REPRESENTATION OF                  MRADØ4ØØ
C        **A(1)*B**(I-2)+A(2)*B**(I-3)+...+A(I-1).                      MRADØ41Ø
         Q = A(I)                                                       MRADØ42Ø
         DO 1 J=1,L                                                     MRADØ43Ø
C           **COMPUTE THE FIRST L COEFFICIENTS OF THE MIXED-            MRADØ44Ø
C           **RADIX REPRESENTATION OF A(I)+B*(C(1)+C(2)*P(1)+           MRADØ45Ø
C           **...+C(L)*P(1)*P(2)*...*P(L-1)).                          MRADØ46Ø
            PP = P(J)                                                   MRADØ47Ø
            QTEMP = Q + B*C(J)                                          MRADØ48Ø
            Q = QTEMP/PP                                                MRADØ49Ø
 1          C(J) = QTEMP - Q*PP                                        MRADØ5ØØ
C        OD                                                             MRADØ51Ø
C        REPEAT ... CONVERT C(1)+C(2)*P(1)+...+C(L)*P(1)*               MRADØ52Ø
C               ... P(2)*...*P(L-1)+Q*P(1)*...*P(L)                     MRADØ53Ø
C               ... TO MIXED-RADIX FORM.                                MRADØ54Ø
 2          IF (Q .NE.Ø)                                    GOTO 3      MRADØ55Ø
                                                            GOTO 4      MRADØ56Ø
C           THEN                                                        MRADØ57Ø
 3             L = L+1                                                  MRADØ58Ø
               IF (L.GT.LMAX) RETURN                                    MRADØ59Ø
               PP = P(L)                                                MRADØ6ØØ
               QTEMP = Q/PP                                             MRADØ61Ø
               C(L) = Q - QTEMP*PP                                      MRADØ62Ø
               Q = QTEMP                                                MRADØ63Ø
                                                            GOTO 2      MRADØ64Ø
C        CONTINUE                                                       MRADØ65Ø
C     OD                                                                MRADØ66Ø
 4                                                          CONTINUEMRADØ67Ø
C     **CONVERT TO SYMMETRIC MIXED-RADIX FORM.                          MRADØ68Ø
        Q = Ø                                                           MRADØ69Ø
        DO 5 I=1,L                                                      MRADØ7ØØ
           C(I) = C(I)+Q                                                MRADØ71Ø
           P1 = (P(I)+1)/2                                              MRADØ72Ø
           Q = C(I)/P1                                                  MRADØ73Ø
 5         C(I) = C(I) - Q*P(I)                                         MRADØ74Ø
C       OD                                                              MRADØ75Ø
        IF (Q.NE.Ø)                                     GOTO 6          MRADØ76Ø
                                                        GOTO 7          MRADØ77Ø
C       THEN                                                            MRADØ78Ø
 6         L = L+1                                                      MRADØ79Ø
           IF (L.GT.LMAX) RETURN                                        MRADØ8ØØ
           C(L) = Q                                                     MRADØ81Ø
 7       IER = Ø                                                        MRADØ82Ø
      RETURN                                                            MRADØ83Ø
      END                                                               MRADØ84Ø

      SUBROUTINE FRADIX(C,N,A,LMAX,L,B,IER)                             FRADØØ1Ø
C                                                                       FRADØØ2Ø
C***************************************************************       FRADØØ3Ø
C GIVEN THE MIXED-RADIX INTEGER                                 *       FRADØØ4Ø
C                                                               *       FRADØØ5Ø
C   C(1)+C(2)*P(1)+ ... +C(N)*P(1)*...*P(N-1),                  *       FRADØØ6Ø
```

```
C                                                                       *       FRADØØ7Ø
C WHERE ABS(C(I)) < (P(I)+1)/2, I=1,...,N,                              *       FRADØØ8Ø
C THIS SUBROUTINE COMPUTES(FOR SOME L <= LMAX) THE COEFFICIENTS*        FRADØØ9Ø
C A(1),A(2),...,A(L) IN ITS FIXED-RADIX REPRESENTATION:                 *       FRADØ1ØØ
C                                                                       *       FRADØ11Ø
C   A(1)*B**(L-1)+A(2)*B*(L-2)+...+A(L),                                *       FRADØ12Ø
C                                                                       *       FRADØ13Ø
C WHERE ABS(A(I)) < B.                                                  *       FRADØ14Ø
C                                                                       *       FRADØ15Ø
C LMAX MUST BE GIVEN SUCH THAT                                          *       FRADØ16Ø
C                                                                       *       FRADØ17Ø
C   LMAX >= (N*LOG(P(N)) - LOG 2) / LOG B.                              *       FRADØ18Ø
C                                                                       *       FRADØ19Ø
C IER IS AN ERROR CODE WHICH IS 1 IF THE DIMENSION PARAMETER N,*        FRADØ2ØØ
C LMAX ARE INCORRECT, AND Ø OTHERWISE.                                  *       FRADØ21Ø
C                                                                       *       FRADØ22Ø
C                    ******* WARNING  *******                           *       FRADØ23Ø
C                                                                       *       FRADØ24Ø
C THIS SUBROUTINE ASSUMMES THAT FOR ALL I                               *       FRADØ25Ø
C                                                                       *       FRADØ26Ø
C    P(I)*B                                                             *       FRADØ27Ø
C                                                                       *       FRADØ28Ø
C DOES NOT OVERFLOW A COMPUTER WORD.                                    *       FRADØ29Ø
C                                                                       *       FRADØ3ØØ
C***********************************************************************       FRADØ31Ø
C                                                                               FRADØ32Ø
      INTEGER C(N),A(LMAX),P(1ØØ),Q,QTEMP,PP,B                                  FRADØ33Ø
      COMMON /PRIMEB/ P, IPRIME(1ØØ)                                            FRAD 34Ø
      IER = 1                                                                   FRADØ35Ø
      IF (N.LT.1 .OR. LMAX.LT.1) RETURN                                         FRADØ36Ø
      L = Ø                                                                     FRADØ37Ø
      DO 5 I=1,N                                                                FRADØ38Ø
C        **AT THIS STAGE A(1)+A(2)*B+...+A(L)*B**(L-1) IS THE                   FRADØ39Ø
C        **FIXED-RADIX REPRESENTATION OF C(N-I+2)+C(N-I+3)*                     FRADØ4ØØ
C        **P(N-I+2)+...+C(N)*P(N-I+2)*...*P(N-1).                               FRADØ41Ø
         NI1 = N-I+1                                                            FRADØ42Ø
         PP = P(NI1)                                                            FRADØ43Ø
         Q = C(NI1)                                                             FRADØ44Ø
         IF (L.GT.Ø)                                          GOTO 1            FRADØ45Ø
                                                              GOTO 3            FRADØ46Ø
C        THEN ... COMPUTE THE FIRST L COEFFICIENTS OF THE                       FRADØ47Ø
C             ... FIXED-RADIX REPRESENTATION OF C(N-I+1)+                       FRADØ48Ø
C             ... P(N-I+1)*(A(1)+A(2)*B+...+A(L)*B**(L-1)).                     FRADØ49Ø
 1          DO 2 J=1,L                                                         FRADØ5ØØ
               QTEMP = A(J)*PP + Q                                              FRADØ51Ø
               Q = QTEMP/B                                                      FRADØ52Ø
 2             A(J) = QTEMP - Q*B                                               FRADØ53Ø
C          OD                                                                   FRADØ54Ø
C        REPEAT ... CONVERT A(1)+A(2)*B+...+A(L)*B**(L-1)                       FRADØ55Ø
C               ... +Q*B**L TO FIXED-RADIX FORM.                                FRADØ56Ø
 3          IF (Q.NE.Ø)                                          GOTO 4            FRADØ57Ø
                                                                 GOTO 5            FRADØ58Ø
C           THEN                                                                 FRADØ59Ø
 4             L = L+1                                                          FRADØ6ØØ
               IF (L.GT.LMAX) RETURN                                            FRADØ61Ø
               QTEMP = Q/B                                                      FRADØ62Ø
               A(L) = Q - QTEMP*B                                               FRADØ63Ø
               Q = QTEMP                                                        FRADØ64Ø
                                                              GOTO 3            FRADØ65Ø
C        CONTINUE                                                               FRADØ66Ø
 5                                                            CONTINUEFRADØ67Ø
C     OD                                                                        FRADØ68Ø
      IF (L.GE.2)                                             GOTO 6            FRADØ69Ø
                                                              GOTO 8            FRADØ7ØØ
C     THEN ... REORDER THE COEFFICIENTS.                                        FRADØ71Ø
 6       L2 = L/2                                                               FRADØ72Ø
         DO 7 I=1,L2                                                            FRADØ73Ø
            LI1 = L-I+1                                                         FRADØ74Ø
            QTEMP = A(I)                                                        FRADØ75Ø
            A(I) = A(LI1)                                                       FRADØ76Ø
 7          A(LI1) = QTEMP                                                      FRADØ77Ø
C        OD                                                                     FRADØ78Ø
 8    IER = Ø                                                                   FRADØ79Ø
      RETURN                                                                    FRADØ8ØØ
      END                                                                       FRADØ81Ø
```

ALGORITHM 523
CONVEX, A New Convex Hull Algorithm for Planar Sets [Z]

WILLIAM F. EDDY
Carnegie-Mellon University

Key Words and Phrases: convex hull, QUICKERSORT, partitioning, sorting
CR Categories: 5.30, 5.31
Language: Fortran

DESCRIPTION

An algorithm, CONVEX, that determines which points of a planar set are vertices of the convex hull of the set is presented here. A detailed explanation of its operation and a report of a small sampling experiment are given in [1].

REFERENCES

1. EDDY, W.F., A new convex hull algorithm for planar sets. *ACM Trans. Math. Software 3*, 4 (Dec. 1977), 398-403).

ALGORITHM

```
      COMMON NCOUNT
      DIMENSION XX(2,25),IN(25),IH(25)
      DIMENSION X(25),Y(25)
      INTEGER IWORK(5Ø)
      INTEGER IL(5Ø)
      DATA IWORK/5Ø*Ø/
      NCOUNT=Ø
C NCOUNT IS TOTAL NUMBER OF POINTS PASSED TO SPLIT
      READ(5,1)N
1     FORMAT(I5)
      WRITE(6,1)N
      N1=N+1
      DO 2 I=1,N
      J=N1-I
2     IN(J)=I
C ARRAY IN CONTAINS INDICES 1-N IN REVERSE ORDER
      DO 3 I=1,N
3     READ(5,4)XX(1,I),XX(2,I)
4     FORMAT(2F1Ø.5)
      DO 5 I=1,N
      J=IN(I)
5     WRITE(6,4)XX(1,J),XX(2,J)
      DO 1Ø M=4,N
      CALL CONVEX(N,XX,M,IN,IWORK,IWORK(N+1),IH,NHULL,IL)
      IK=IL(1)
```

Received 24 September 1975; revised 25 June 1976 and 12 January 1977.

This research was supported in part by the National Science Foundation under Grant DCR75-08374.
Author's address: Department of Statistics, Carnegie-Mellon University, Schenley Park, Pittsburgh, PA 15213.

```
      DO 6 I=1,NHULL
      J=IH(IK)
      X(I)=XX(1,J)
      Y(I)=XX(2,J)
6     IK=IL(IK)
      WRITE(6,7)M,NHULL,NCOUNT
7     FORMAT(12HØSAMPLE SIZE ,I5,9H VERTICES ,I5,6H SPLIT ,I5)
      DO 8 I=1,NHULL
8     WRITE(6,9)X(I),Y(I)
9     FORMAT(1X,2F1Ø.5)
1Ø    CONTINUE
      STOP
      END


      SUBROUTINE SPLIT(N,X,M,IN,II,JJ,S,IABV,NA,MAXA,IBEL,
     1  NB,MAXB)
C THIS SUBROUTINE TAKES THE M POINTS OF ARRAY X WHOSE
C SUBSCRIPTS ARE IN ARRAY IN AND PARTITIONS THEM BY THE
C LINE JOINING THE TWO POINTS IN ARRAY X WHOSE SUBSCRIPTS
C ARE II AND JJ. THE SUBSCRIPTS OF THE POINTS ABOVE THE
C LINE ARE PUT INTO ARRAY IABV, AND THE SUBSCRIPTS OF THE
C POINTS BELOW ARE PUT INTO ARRAY IBEL. NA AND NB ARE,
C RESPECTIVELY, THE NUMBER OF POINTS ABOVE THE LINE AND THE
C NUMBER BELOW. MAXA AND MAXB ARE THE SUBSCRIPTS FOR ARRAY
C X OF THE POINT FURTHEST ABOVE THE LINE AND THE POINT
C FURTHEST BELOW, RESPECTIVELY. IF EITHER SUBSET IS NULL
C THE CORRESPONDING SUBSCRIPT (MAXA OR MAXB) IS SET TO ZERO
C FORMAL PARAMETERS
C INPUT
C N     INTEGER            TOTAL NUMBER OF DATA POINTS
C X     REAL ARRAY (2,N)   (X,Y) CO-ORDINATES OF THE DATA
C M     INTEGER            NUMBER OF POINTS IN INPUT SUBSET
C IN    INTEGER ARRAY (M)  SUBSCRIPTS FOR ARRAY X OF THE
C                          POINTS IN THE INPUT SUBSET
C II    INTEGER            SUBSCRIPT FOR ARRAY X OF ONE POINT
C                          ON THE PARTITIONING LINE
C JJ    INTEGER            SUBSCRIPT FOR ARRAY X OF ANOTHER
C                          POINT ON THE PARTITIONING LINE
C S     INTEGER            SWITCH TO DETERMINE OUTPUT. REFER
C                          TO COMMENTS BELOW
C OUTPUT
C IABV INTEGER ARRAY (M)   SUBSCRIPTS FOR ARRAY X OF THE
C                          POINTS ABOVE THE PARTITIONING LINE
C NA    INTEGER            NUMBER OF ELEMENTS IN IABV
C MAXA INTEGER             SUBSCRIPT FOR ARRAY X OF POINT
C                          FURTHEST ABOVE THE LINE. SET TO
C                          ZERO IF NA IS ZERO
C IBEL INTEGER ARRAY (M)   SUBSCRIPTS FOR ARRAY X OF THE
C                          POINTS BELOW THE PARTITIONING LINE
C NB    INTEGER            NUMBER OF ELEMENTS IN IBEL
C MAXB INTEGER             SUBSCRIPT FOR ARRAY X OF PCINT
C                          FURTHEST BELOW THE LINE. SET TO
C                          ZERO IF NB IS ZERO
      DIMENSION X(2,N)
      DIMENSION IN(M),IABV(M),IBEL(M)
      INTEGER S
C IF S = 2 DONT SAVE IBEL,NB,MAXB.
C IF S =-2 DONT SAVE IABV,NA,MAXA.
C OTHERWISE SAVE EVERYTHING
C IF S IS POSITIVE THE ARRAY BEING PARTITIONED IS ABOVE
C THE INITIAL PARTITIONING LINE. IF IT IS NEGATIVE, THEN
C THE SET OF POINTS IS BELOW.
      LOGICAL T
      T=.FALSE.
C CHECK TO SEE IF THE LINE IS VERTICAL
      IF(X(1,JJ).NE.X(1,II))GOTO 1
      XT=X(1,II)
      DIR=SIGN(1.,X(2,JJ)-X(2,II))*SIGN(1.,FLOAT(S))
      T=.TRUE.
      GOTO 2
1     A=(X(2,JJ)-X(2,II))/(X(1,JJ)-X(1,II))
      B=X(2,II)-A*X(1,II)
2     UP=Ø.
      NA=Ø
      MAXA=Ø
      DOWN=Ø.
```

```
        NB=Ø
        MAXB=Ø
        DO 6 I=1,M
          IS=IN(I)
        IF(T)GOTO 3
        Z=X(2,IS)-A*X(1,IS)-B
        GOTO 4
3       Z=DIR*(X(1,IS)-XT)
4       IF(Z.LE.Ø.)GOTO 5
C THE POINT IS ABOVE THE LINE
        IF(S.EQ.-2)GOTO 6
        NA=NA+1
        IABV(NA)=IS
        IF(Z.LT.UP)GOTO 6
        UP=Z
        MAXA=NA
        GOTO 6
5       IF(S.EQ.2)GOTO 6
        IF(Z.GE.Ø.)GOTO 6
C THE POINT IS BELOW THE LINE
        NB=NB+1
        IBEL(NB)=IS
        IF(Z.GT.DOWN)GOTO 6
        DOWN=Z
        MAXB=NB
6     CONTINUE
      RETURN
      END


      SUBROUTINE CONVEX(N,X,M,IN,IA,IB,IH,NH,IL)
C THIS SUBROUTINE DETERMINES WHICH OF THE M POINTS OF ARRAY
C X WHOSE SUBSCRIPTS ARE IN ARRAY IN ARE VERTICES OF THE
C MINIMUM AREA CONVEX POLYGON CONTAINING THE M POINTS. THE
C SUBSCRIPTS OF THE VERTICES ARE PLACED IN ARRAY IH IN THE
C ORDER THEY ARE FOUND. NH IS THE NUMBER OF ELEMENTS IN
C ARRAY IH AND ARRAY IL. ARRAY IL IS A LINKED LIST GIVING
C THE ORDER OF THE ELEMENTS OF ARRAY IH IN A COUNTER
C CLOCKWISE DIRECTION. THIS ALGORITHM CORRESPONDS TO A
C PREORDER TRAVERSAL OF A CERTAIN BINARY TREE. EACH VERTEX
C OF THE BINARY TREE REPRESENTS A SUBSET OF THE M POINTS.
C AT EACH STEP THE SUBSET OF POINTS CORRESPONDING TO THE
C CURRENT VERTEX OF THE TREE IS PARTITIONED BY A LINE
C JOINING TWO VERTICES OF THE CONVEX POLYGON. THE LEFT SON
C VERTEX IN THE BINARY TREE REPRESENTS THE SUBSET OF POINTS
C ABOVE THE PARTITIONING LINE AND THE RIGHT SON VERTEX, THE
C SUBSET BELOW THE LINE. THE LEAVES OF THE TREE REPRESENT
C EITHER NULL SUBSETS OR SUBSETS INSIDE A TRIANGLE WHOSE
C VERTICES COINCIDE WITH VERTICES OF THE CONVEX POLYGON.
C FORMAL PARAMETERS
C INPUT
C N  INTEGER            TOTAL NUMBER OF DATA POINTS
C X  REAL ARRAY (2,N)   (X,Y) CO-ORDINATES OF THE DATA
C M  INTEGER            NUMBER OF POINTS IN THE INPUT SUBSET
C IN INTEGER ARRAY (M)  SUBSCRIPTS FOR ARRAY X OF THE POINTS
C                       IN THE INPUT SUBSET
C WORK AREA
C IA INTEGER ARRAY (M)  SUBSCRIPTS FOR ARRAY X OF LEFT SON
C                       SUBSETS. SEE COMMENTS AFTER DIMENSION
C                       STATEMENTS
C IB INTEGER ARRAY (M)  SUBSCRIPTS FOR ARRAY X OF RIGHT SON
C                       SUBSETS
C OUTPUT
C IH INTEGER ARRAY (M)  SUBSCRIPTS FOR ARRAY X OF THE
C                       VERTICES OF THE CONVEX HULL
C NH INTEGER            NUMBER OF ELEMENTS IN ARRAY IH AND
C                       ARRAY IL. SAME AS NUMBER OF VERTICES
C                       OF THE CONVEX POLYGON
C IL INTEGER ARRAY (M)  A LINKED LIST GIVING IN ORDER IN A
C                       COUNTER-CLOCKWISE DIRECTION THE
C                       ELEMENTS OF ARRAY IH
      DIMENSION X(2,N)
      DIMENSION IN(M),IA(M),IB(M),IH(M),IL(M)
C THE UPPER END OF ARRAY IA IS USED TO STORE TEMPORARILY
C THE SIZES OF THE SUBSETS WHICH CORRESPOND TO RIGHT SON
C VERTICES, WHILE TRAVERSING DOWN THE LEFT SONS WHEN ON THE
C LEFT HALF OF THE TREE, AND TO STORE THE SIZES OF THE LEFT
```

```
C SONS WHILE TRAVERSING THE RIGHT SONS(DOWN THE RIGHT HALF)
      LOGICAL MAXE,MINE
      IF(M.EQ.1)GOTO 22
      IL(1)=2
      IL(2)=1
      KN=IN(1)
      KX=IN(2)
      IF(M.EQ.2)GOTO 21
      MP1=M+1
      MIN=1
      MX=1
      KX=IN(1)
      MAXE=.FALSE.
      MINE=.FALSE.
C FIND TWO VERTICES OF THE CONVEX HULL FOR THE INITIAL
C PARTITION
      DO 6 I=2,M
        J=IN(I)
        IF(X(1,J)-X(1,KX))3,1,2
1       MAXE=.TRUE.
        GOTO 3
2       MAXE=.FALSE.
        MX=I
        KX=J
3       IF(X(1,J)-X(1,KN))5,4,6
4       MINE=.TRUE.
        GOTO 6
5       MINE=.FALSE.
        MIN=I
        KN=J
6     CONTINUE
C IF THE MAX AND MIN ARE EQUAL, ALL M POINTS LIE ON A
C VERTICAL LINE
      IF(KX.EQ.KN)GOTO 18
C IF MAXE (OR MINE) HAS THE VALUE TRUE THERE ARE SEVERAL
C MAXIMA (OR MINIMA) WITH EQUAL FIRST COORDINATES
      IF(MAXE.OR.MINE)GOTO 23
7     IH(1)=KX
      IH(2)=KN
      NH=3
      INH=1
      NIB=1
      MA=M
      IN(MX)=IN(M)
      IN(M)=KX
      MM=M-2
      IF(MIN.EQ.M)MIN=MX
      IN(MIN)=IN(M-1)
      IN(M-1)=KN
C BEGIN BY PARTITIONING THE ROOT OF THE TREE
      CALL SPLIT(N,X,MM,IN,IH(1),IH(2),Ø,IA,MB,MXA,IB,IA(MA),
     1  MXBB)
C FIRST TRAVERSE THE LEFT HALF OF THE TREE
C START WITH THE LEFT SON
8     NIB=NIB+IA(MA)
      MA=MA-1
9     IF(MXA.EQ.Ø)GOTO 11
      IL(NH)=IL(INH)
      IL(INH)=NH
      IH(NH)=IA(MXA)
      IA(MXA)=IA(MB)
      MB=MB-1
      NH=NH+1
      IF(MB.EQ.Ø)GOTO 1Ø
      ILINH=IL(INH)
      CALL SPLIT(N,X,MB,IA,IH(INH),IH(ILINH),1,IA,MBB,MXA,
     1  IB(NIB),IA(MA),MXB)
      MB=MBB
      GOTO 8
C THEN THE RIGHT SON
1Ø    INH=IL(INH)
11    INH=IL(INH)
      MA=MA+1
      NIB=NIB-IA(MA)
      IF(MA.GE.M)GOTO 12
      IF(IA(MA).EQ.Ø)GOTO 11
      ILINH=IL(INH)
```

```
C ON THE LEFT SIDE OF THE TREE, THE RIGHT SON OF A RIGHT SON
C MUST REPRESENT A SUBSET OF POINTS WHICH IS INSIDE A
C TRIANGLE WITH VERTICES WHICH ARE ALSO VERTICES OF THE
C CONVEX POLYGON AND HENCE THE SUBSET MAY BE NEGLECTED.
      CALL SPLIT(N,X,IA(MA),IB(NIB),IH(INH),IH(ILINH),2,IA,
     1  MB,MXA,IB(NIB),MBB,MXB)
      IA(MA)=MBB
      GOTO 9
C NOW TRAVERSE THE RIGHT HALF OF THE TREE
12    MXB=MXBB
      MA=M
      MB=IA(MA)
      NIA=1
      IA(MA)=Ø
C START WITH THE RIGHT SON
13    NIA=NIA+IA(MA)
      MA=MA-1
14    IF(MXB.EQ.Ø)GOTO 16
      IL(NH)=IL(INH)
      IL(INH)=NH
      IH(NH)=IB(MXB)
      IB(MXB)=IB(MB)
      MB=MB-1
      NH=NH+1
      IF(MB.EQ.Ø)GOTO 15
      ILINH=IL(INH)
      CALL SPLIT(N,X,MB,IB(NIB),IH(INH),IH(ILINH),-1,IA(NIA),
     1  IA(MA),MXA,IB(NIB),MBB,MXB)
      MB=MBB
      GOTO 13
C THEN THE LEFT SON
15    INH=IL(INH)
16    INH=IL(INH)
      MA=MA+1
      NIA=NIA-IA(MA)
      IF(MA.EQ.MP1)GOTO 17
      IF(IA(MA).EQ.Ø)GOTO 16
      ILINH=IL(INH)
C ON THE RIGHT SIDE OF THE TREE, THE LEFT SON OF A LEFT SON
C MUST REPRESENT A SUBSET OF POINTS WHICH IS INSIDE A
C TRIANGLE WITH VERTICES WHICH ARE ALSO VERTICES OF THE
C CONVEX POLYGON AND HENCE THE SUBSET MAY BE NEGLECTED.
      CALL SPLIT(N,X,IA(MA),IA(NIA),IH(INH),IH(ILINH),-2,
     1  IA(NIA),MBB,MXA,IB(NIB),MB,MXB)
      GOTO 14
17    NH=NH-1
      RETURN
C ALL THE SPECIAL CASES ARE HANDLED DOWN HERE
C IF ALL THE POINTS LIE ON A VERTICAL LINE
18    KX=IN(1)
      KN=IN(1)
      DO 2Ø I=1,M
        J=IN(I)
        IF(X(2,J).LE.X(2,KX))GOTO 19
        MX=I
        KX=J
19      IF(X(2,J).GE.X(2,KN))GOTO 2Ø
        MIN=I
        KN=J
2Ø    CONTINUE
      IF(KX.EQ.KN)GOTO 22
C IF THERE ARE ONLY TWO POINTS
21    IH(1)=KX
      IH(2)=KN
      NH=3
      IF((X(1,KN).EQ.X(1,KX)).AND.(X(2,KN).EQ.X(2,KX)))NH=2
      GOTO 17
C IF THERE IS ONLY ONE POINT
22    NH=2
      IH(1)=IN(1)
      IL(1)=1
      GOTO 17
C MULTIPLE EXTREMES ARE HANDLED HERE
C IF THERE ARE SEVERAL POINTS WITH THE (SAME) LARGEST
C FIRST COORDINATE
23    IF(.NOT.MAXE)GOTO 25
```

```
      DO 24 I=1,M
        J=IN(I)
        IF(X(1,J).NE.X(1,KX))GOTO 24
        IF(X(2,J).LE.X(2,KX))GOTO 24
        MX=I
        KX=J
24    CONTINUE
C IF THERE ARE SEVERAL POINTS WITH THE (SAME) SMALLEST
C FIRST COORDINATE
25    IF(.NOT.MINE)GOTO 7
      DO 26 I=1,M
        J=IN(I)
        IF(X(1,J).NE.X(1,KN))GOTO 26
        IF(X(2,J).GE.X(2,KN))GOTO 26
        MIN=I
        KN=J
26    CONTINUE
      GOTO 7
      END
   2Ø
2.Ø       Ø.Ø
1.73      -1.Ø
1.Ø       1.73
Ø.Ø       2.Ø
Ø.1       Ø.1
-1.Ø      -1.73
Ø.2       -Ø.2
-1.73     1.Ø
-Ø.3      Ø.3
Ø.Ø       -2.Ø
-Ø.4      -Ø.4
-2.Ø      Ø.Ø
Ø.5       Ø.5
1.73      1.Ø
Ø.6       -Ø.6
-1.Ø      1.73
-Ø.7      Ø.7
-1.73     -1.Ø
-Ø.8      -Ø.8
1.Ø       -1.73
```

ALGORITHM 524
MP, A Fortran Multiple-Precision Arithmetic Package [A1]

RICHARD P. BRENT
Australian National University

Key Words and Phrases: arithmetic, multiple precision, extended precision, floating point, elementary function evaluation, Euler's constant, gamma function, polyalgorithm, software package, Fortran, machine-independent software, special function evaluation, Bessel functions, exponential integral, logarithmic intergral, Bernoulli numbers, zeta function, portable software
CR Categories: 3.15, 4.49, 5.11, 5.12, 5.15, 5.19, 5.25
Language: Fortran

DESCRIPTION

The design of the package and the theoretical background for the algorithms used are given in [1]. Details of calling sequences, etc., are given in the comments included here and in [2].

REFERENCES

1. BRENT, R.P. A Fortran multiple-precision arithmetic package. *ACM Trans. Math. Software 4*, 1 (March 1978), 57–70.
2. BRENT, R.P. MP users guide. Tech. Rep. 54, Computer Centre, Australian National U., Canberra, Australia, Sept. 1976.

ALGORITHM

[Only that portion of the listing which gives the introductory comments and a small example program is printed here. The complete listing, together with a Users' Guide giving further details, is available from the ACM Algorithms Distribution Service.

```
C ABCDEFGHIJKLMNOPQRSTUVWXYZ$Ø123456789+-*/=(),.                    MPØØØØ1Ø
C                                                                   MPØØØØ2Ø
C $$                    ******  COMMENTS  ******                   MPØØØØ3Ø
C                                                                   MPØØØØ4Ø
C DESCRIPTION OF MP (VERSION OF 17 FEBRUARY 1977)                   MPØØØØ51
C *****************                                                 MPØØØØ6Ø
C                                                                   MPØØØØ7Ø
C MP IS A MULTIPLE-PRECISION ARITHMETIC PACKAGE.                    MPØØØØ8Ø
C IT IS ALMOST COMPLETELY MACHINE-INDEPENDENT, AND SHOULD           MPØØØØ9Ø
```

Received July 1975; revised October 1975, March 1976, and October 1976.

Author's address: Computer Centre, Australian National University, Box 4, Canberra, ACT 2600, Australia.

```
C RUN ON ANY MACHINE WITH AN ANSI STANDARD FORTRAN COMPILER,          MP000100
C SUFFICIENT MEMORY, AND A WORDLENGTH OF AT LEAST 16 BITS.            MP000110
C SOME MODIFICATIONS WOULD BE NECESSARY FOR A WORDLENGTH              MP000120
C OF LESS THAN 16 BITS.                                               MP000130
C                                                                     MP000140
C MP WAS WRITTEN BY R. BRENT (COMPUTER CENTRE, AUSTRALIAN             MP000150
C NATIONAL UNIVERSITY), IN NOVEMBER 1973 (VERSION 1).                 MP000160
C CORRESPONDENCE SHOULD BE SENT TO R. P. BRENT, COMPUTER              MP000170
C CENTRE, ANU, BOX 4, CANBERRA, ACT 2600, AUSTRALIA.                  MP000180
C                                                                     MP000190
C MP HAS BEEN TESTED ON A UNIVAC 1108  (E LEVEL FORTRAN V),           MP000200
C A UNIVAC 1100/42 (E AND T LEVEL FORTRAN V, ASCII FORTRAN,           MP000210
C AND RALPH), A PDP 10 (FORTRAN 10(/NOOPT) AND FORTRAN 40),           MP000220
C AN IBM 360/50 (FORTRAN G AND FORTRAN H, OPT = 2),                   MP000230
C AN IBM 360/91 AND 370/168 (FORTRAN H EXTENDED, OPT = 2),            MP000240
C A CYBER 76 (FTN 4.2, OPT = 1) AND A PDP 11/45 (DOS).                MP000250
C THESE MACHINES HAVE EFFECTIVE INTEGER WORDLENGTHS RANGING           MP000260
C FROM 16 TO 48 BITS.                                                 MP000270
C                                                                     MP000280
C MP WORKS WITH NORMALIZED FLOATING-POINT NUMBERS.                    MP000290
C THE BASE (B) AND NUMBER OF DIGITS (T) ARE                           MP000300
C ARBITRARY (SUBJECT TO SOME RESTRICTIONS GIVEN BELOW),               MP000310
C AND MAY BE VARIED DYNAMICALLY.                                      MP000320
C                                                                     MP000330
C T-DIGIT FLOATING-POINT NUMBERS ARE STORED IN INTEGER ARRAYS OF      MP000340
C DIMENSION T+2, WITH THE FOLLOWING CONVENTIONS -                     MP000350
C   WORD 1 = SIGN (0, -1 OR +1)                                       MP000360
C   WORD 2 = EXPONENT (TO BASE B)                                     MP000370
C   WORDS 3 TO T+2 = NORMALIZED FRACTION                              MP000380
C NOTE THAT WORDS 2 TO T+2 ARE UNDEFINED IF SIGN = 0.                 MP000390
C                                                                     MP000400
C ARITHMETIC IS ROUNDED, AND FOUR GUARD DIGITS ARE USED               MP000410
C FOR ADDITION AND MULTIPLICATION, SO THE CORRECTLY ROUNDED           MP000420
C RESULT IS USUALLY PRODUCED.   DIVISION, SQRT ETC ARE DONE           MP000430
C BY NEWTONS METHOD, BUT GIVE THE EXACT RESULT IF IT CAN BE           MP000440
C REPRESENTED WITH T-2 DIGITS. OTHER ROUTINES (MPSIN, MPLN ETC)       MP000450
C USUALLY GIVE A RESULT Y = F(X) WHICH COULD BE OBTAINED              MP000460
C BY MAKING AN O(B**(1-T)) PERTURBATION IN X, EVALUATING F            MP000470
C EXACTLY, THEN MAKING AN O(B**(1-T)) PERTURBATION IN Y.              MP000480
C                                                                     MP000490
C EXPONENTS CAN LIE IN THE RANGE -M, ... , +M INCLUSIVE,              MP000500
C WHERE M IS SET BY THE USER.                                         MP000510
C ON UNDERFLOW DURING AN ARITHMETIC OPERATION, THE RESULT             MP000520
C IS SET TO ZERO BY SUBROUTINE MPUNFL.                                MP000530
C ON OVERFLOW SUBROUTINE MPOVFL IS CALLED AND EXECUTION               MP000540
C IS TERMINATED WITH AN ERROR MESSAGE.                                MP000550
C ERROR MESSAGES ARE PRINTED ON LOGICAL UNIT LUN, WHERE LUN           MP000560
C IS SET BY THE USER, AND THEN EXECUTION IS TERMINATED                MP000570
C BY A CALL TO SUBROUTINE MPERR.  IT IS ASSUMED THAT LOGICAL          MP000580
C RECORDS OF UP TO 80 CHARACTERS MAY BE WRITTEN ON UNIT LUN.          MP000590
C A WORKING ARRAY OF SIZE MXR (SEE BELOW) MUST BE PROVIDED            MP000600
C IN COMMON.                                                          MP000610
C                                                                     MP000620
C THE PARAMETERS B, T, M, LUN AND MXR ARE PASSED TO THE UTILITY       MP000630
C ROUTINES IN COMMON, TOGETHER WITH A WORKING ARRAY R WHICH           MP000640
C MUST BE SUFFICIENTLY LARGE (SEE BELOW).  MOST ROUTINES              MP000650
C USE THE STATEMENTS -                                                MP000660
C      COMMON B, T, M, LUN, MXR, R                                    MP000670
C      INTEGER B, T, R(1)                                             MP000680
C AND IT IS ASSUMED THAT R IS DIMENSIONED SUFFICIENTLY LARGE          MP000690
C IN THE CALLING PROGRAM, AND THAT MXR IS SET TO THE                  MP000700
C DIMENSION OF R IN THE CALLING PROGRAM.                              MP000710
C WARNING - IT IS ASSUMED THAT THE COMPILER PASSES ADDRESSES OF       MP000720
C *******   ARRAYS USED AS ARGUMENTS IN SUBROUTINE CALLS (I.E., CALL  MP000730
C           BY REFERENCE), AND DOES NOT CHECK FOR ARRAY BOUNDS        MP000740
C           VIOLATIONS (EITHER FOR ARGUMENTS OR FOR ARRAYS IN COMMON). MP000750
C           APART FROM THESE VIOLATIONS, MP IS WRITTEN ENTIRELY IN    MP000760
C           ANSI STANDARD FORTRAN (ANSI X3.9-1966).  THIS HAS BEEN    MP000770
C           CHECKED BY THE PFORT VERIFIER.                            MP000780
C                                                                     MP000790
C RESTRICTIONS -                                                      MP000800
C      B (THE BASE) MUST BE AT LEAST 2,                               MP000810
C      T (NUMBER OF DIGITS) MUST BE AT LEAST 2,                       MP000820
C      M (EXPONENT RANGE) MUST BE GREATER THAN T AND LESS             MP000830
C        THAN 1/4 THE LARGEST MACHINE-REPRESENTABLE INTEGER,          MP000840
C      8*B**2-1  MUST BE NO GREATER THAN THE LARGEST MACHINE-         MP000850
```

```
C        REPRESENTABLE INTEGER, AND THE INTEGERS Ø, 1, ... , B          MPØØØ86Ø
C        MUST BE EXACTLY REPRESENTABLE AS SINGLE-PRECISION FLOATING-    MPØØØ87Ø
C        POINT NUMBERS, AND                                             MPØØØ88Ø
C      B**(T-1) SHOULD BE AT LEAST 1Ø**7.                               MPØØØ89Ø
C                                                                       MPØØØ9ØØ
C B AND T MAY BE SET TO GIVE THE EQUIVALENT OF A SPECIFIED              MPØØØ91Ø
C NUMBER OF DECIMAL PLACES BY CALLING MPSET (SEE BELOW), OR MAY         MPØØØ92Ø
C BE SET DIRECTLY BY THE USER.  IF MPSET IS NOT CALLED, THE USER        MPØØØ93Ø
C MUST REMEMBER TO INITIALIZE M, LUN AMD MXR (SEE ABOVE) AS WELL        MPØØØ94Ø
C AS B AND T BEFORE CALLING ANY MP ROUTINES.                            MPØØØ95Ø
C                                                                       MPØØØ96Ø
C FOR EFFICIENCY CHOOSE B FAIRLY LARGE, SUBJECT TO THE RESTRICTIONS     MPØØØ97Ø
C GIVEN ABOVE.  FOR EXAMPLE, IF THE WORDLENGTH IS                       MPØØØ98Ø
C      48 BITS, COULD USE B = 4194304 OR 1ØØØØØØ,                       MPØØØ99Ø
C      36 BITS, COULD USE B = 65536 OR 1ØØØØ,                           MPØØ1ØØØ
C      32 BITS, COULD USE B = 16384 OR 1ØØØØ,                           MPØØ1Ø1Ø
C      24 BITS, COULD USE B = 1Ø24 OR 1ØØØ,                             MPØØ1Ø2Ø
C      18 BITS, COULD USE B = 128 OR 1ØØ,                               MPØØ1Ø3Ø
C      16 BITS, COULD USE B = 64 OR 1Ø.                                 MPØØ1Ø4Ø
C                                                                       MPØØ1Ø5Ø
C AVOID MULTIPLICATION OR DIVISION BY MP NUMBERS, AS                    MPØØ1Ø6Ø
C THESE TAKE O(T**2) OPERATIONS, WHEREAS MULT./DIV. BY                  MPØØ1Ø7Ø
C INTEGERS TAKE O(T) OPERATIONS.                                        MPØØ1Ø8Ø
C                                                                       MPØØ1Ø9Ø
C MP NUMBERS USED AS ARGUMENTS OF SUBROUTINES NEED NOT BE               MPØØ11ØØ
C DISTINCT.  FOR EXAMPLE,                                               MPØØ111Ø
C       CALL MPADD (X, Y, Y)  OR  CALL MPEXP (X, X) ARE OK.            MPØØ112Ø
C HOWEVER, DISTINCT ARRAYS WHICH OVERLAP SHOULD NOT BE USED.            MPØØ113Ø
C                                                                       MPØØ114Ø
C FOR ADDITIONAL DETAILS SEE - A FORTRAN MULTIPLE-PRECISION             MPØØ115Ø
C ARITHMETIC PACKAGE (BY R. P. BRENT), TO APPEAR IN ACM                 MPØØ116Ø
C TRANSACTIONS ON MATHEMATICAL SOFTWARE (AVAILABLE AS A                 MPØØ117Ø
C CARNEGIE-MELLON UNIV. COMPUTER SCIENCE DEPT. REPORT,                  MPØØ118Ø
C PITTSBURGH, PENNSYLVANIA, MAY 1976) AND THE MP USERS GUIDE.           MPØØ119Ø
C                                                                       MPØØ12ØØ
C SUMMARY OF MP ROUTINES                                                MPØØ121Ø
C **********************                                                MPØØ122Ø
C                                                                       MPØØ123Ø
C BASIC ARITHMETIC - MPADD, MPADDI, MPADDQ, MPDIV, MPDIVI,              MPØØ124Ø
C                    MPMUL, MPMULI, MPMULQ, MPREC, MPSUB                MPØØ125Ø
C                                                                       MPØØ126Ø
C POWERS AND ROOTS - MPPWR, MPPWR2, MPQPWR, MPROOT, MPSQRT              MPØØ127Ø
C                                                                       MPØØ128Ø
C ELEMENTARY FUNCTIONS - MPASIN, MPATAN, MPCOS, MPCOSH, MPEXP,          MPØØ129Ø
C                    MPLN, MPLNGS, MPLNI, MPSIN, MPSINH, MPTAN,         MPØØ13ØØ
C                    MPTANH                                             MPØØ131Ø
C                                                                       MPØØ132Ø
C SPECIAL FUNCTIONS - MPBESJ, MPDAW, MPEI, MPERF, MPERFC,               MPØØ133Ø
C                    MPGAM, MPGAMQ, MPLI, MPLNGM                        MPØØ134Ø
C                                                                       MPØØ135Ø
C CONSTANTS - MPBERN, MPEPS, MPEUL, MPMAXR, MPMINR, MPPI,               MPØØ136Ø
C                    MPPIGL, MPZETA                                     MPØØ137Ø
C                                                                       MPØØ138Ø
C INPUT AND OUTPUT - MPDUMP, MPIN, MPINE, MPOUT, MPOUTE, MPOUT2        MPØØ139Ø
C                                                                       MPØØ14ØØ
C CONVERSION - MPCDM, MPCIM, MPCMD, CPCMDE, MPCMEF, MPCMI,             MPØØ141Ø
C                    MPCMIM, MPCMR, MPCMRE, MPCQM, MPCRM                MPØØ142Ø
C                                                                       MPØØ143Ø
C COMPARISON - MPCMPA, MPCMPI, MPCMPR, MPCOMP                           MPØØ144Ø
C                                                                       MPØØ145Ø
C GENERAL UTILITY ROUTINES - MPABS, MPCLR, MPCMF, MPMAX, MPMIN,         MPØØ146Ø
C                    MPNEG, MPPACK, MPPOLY, MPSET, MPSTR, MPUNPK        MPØØ147Ø
C                                                                       MPØØ148Ø
C ERROR DETECTION AND HANDLING - MPCHK, MPERR, MPOVFL, MPUNFL           MPØØ149Ø
C                                                                       MPØØ15ØØ
C TEST PROGRAMS - EXAMPLE, TEST, TESTV, TEST2                           MPØØ151Ø
C                                                                       MPØØ152Ø
C MISCELLANEOUS ROUTINES USED BY THE ABOVE - MPADD2, MPADD3,            MPØØ153Ø
C                    MPART1, MPBES2, MPERF2, MPERF3, MPEXP1,            MPØØ154Ø
C                    MPEXT, MPGCD, MPHANK, MPLNS, MPL235,               MPØØ155Ø
C                    MPMLP, MPMUL2, MPNZR, MPSIN1, MP4ØD,               MPØØ156Ø
C                    MP4ØE, MP4ØF, MP4ØG, TIMEMP                        MPØØ157Ø
C                                                                       MPØØ158Ø
C LIST OF MP ROUTINES                                                   MPØØ159Ø
C ********************                                                 MPØØ16ØØ
C                                                                       MPØØ161Ø
```

```
C THE ROUTINES PROVIDED IN MP ARE LISTED BELOW.  FOR MORE DETAILS OF    MP00162Ø
C CALLING SEQUENCES, RESTRICTIONS, ACCURACY, AND ERROR CONDITIONS,      MP00163Ø
C SEE THE COMMENTS IN EACH ROUTINE.  SPACE REQUIRED (I.E. DIMENSION     MP00164Ø
C OF R IN COMMON) IS T+4 WORDS UNLESS NOTED BELOW.  THE ROUTINES        MP00165Ø
C INDICATED BY * ARE THOSE MOST LIKELY TO BE OF INTEREST TO THE USER.   MP00166Ø
C FOR FAST EXECUTION THE ROUTINES WHICH SHOULD BE OPTIMIZED ARE         MP00167Ø
C MPNZR, MPMLP, MPDIVI, MPADD2, MPADD3, AND MPMUL2.                     MP00168Ø
C                                                                       MP00169Ø
C IN THE LIST BELOW AN MP NUMBER MEANS A MULTIPLE-PRECISION             MP00170Ø
C NUMBER AS DESCRIBED ABOVE, M(T) MEANS THE TIME TO MULTIPLY TWO        MP00171Ø
C T-DIGIT MP NUMBERS (SEE MPMUL), AN INTEGER MEANS A SINGLE-PRECISION   MP00172Ø
C INTEGER, A RATIONAL NUMBER MEANS THE RATIO OF TWO INTEGERS.           MP00173Ø
C X, Y, ... ARE MP NUMBERS,  I, J, ... ARE INTEGERS,                    MP00174Ø
C AND RI, RX, ... ARE SINGLE-PRECISION REAL NUMBERS.                    MP00175Ø
C TIME BOUNDS SUCH AS O(T**2) ARE AS T TENDS TO INFINITY                MP00176Ø
C WITH EVERYTHING ELSE FIXED.                                           MP00177Ø
C                                                                       MP00178Ø
C *  EXAMPLE   A SMALL MAIN PROGRAM GIVING AN EXAMPLE OF THE USE OF MP.  MP00179Ø
C *  MPABS     COMPUTES ABSOLUTE VALUE OF AN MP NUMBER                   MP00180Ø
C              CALL MPABS (X, Y) MEANS Y = ABS(X)                        MP00181Ø
C *  MPADD     ADDS TWO MP NUMBERS                                       MP00182Ø
C              CALL MPADD (X, Y, Z) MEANS Z = X + Y                      MP00183Ø
C *  MPADDI    ADDS AN MP NUMBER TO AN INTEGER,                          MP00184Ø
C              GIVING A MULTIPLE-PRECISION RESULT, SPACE = 2T+6          MP00185Ø
C              CALL MPADDI (X, IY, Z) MEANS Z = X + IY                   MP00186Ø
C *  MPADDQ    ADDS A RATIONAL NUMBER TO AN MP NUMBER,                   MP00187Ø
C              SPACE = 2T+6                                              MP00188Ø
C              CALL MPADDQ (X, I, J, Y) MEANS Y = X + I/J                MP00189Ø
C    MPADD2    ROUTINE CALLED BY MPADD AND MPSUB                         MP00190Ø
C    MPADD3    ROUTINE CALLED BY MPADD2                                  MP00191Ø
C    MPART1    COMPUTES ARCTAN(1/N) FOR N .GT. 1 (CALLED BY MPPI)        MP00192Ø
C              SPACE = 2T+6                                              MP00193Ø
C *  MPASIN    COMPUTES ARCSIN OF AN MP NUMBER,                          MP00194Ø
C              USING AN O(M(T)T) METHOD, SPACE = 5T+12                   MP00195Ø
C              CALL MPASIN (X, Y) MEANS Y = ARCSIN(X)                    MP00196Ø
C *  MPATAN    COMPUTES ARCTAN OF AN MP NUMBER                           MP00197Ø
C              USING AN O(T.M(T)) ALGORITHM, SPACE = 5T+12               MP00198Ø
C              CALL MPATAN (X, Y) MEANS Y = ARCTAN(X)                    MP00199Ø
C *  MPBERN    COMPUTES BERNOULLI NUMBERS B2, B4, B6, ...                MP00200Ø
C              SPACE = 8T+18                                             MP00201Ø
C *  MPBESJ    COMPUTES BESSEL FUNCTION J(NU,X) FOR MP X                 MP00202Ø
C              AND SMALL INTEGER NU, SPACE = 14T+156                     MP00203Ø
C              CALL MPBESJ (X, NU, Y) MEANS Y = J(NU,X)                  MP00204Ø
C    MPBES2    ROUTINE CALLED BY MPBESJ (USES BACKWARD RECURRENCE        MP00205Ø
C              TO EVALUATE J(NU,X)), SPACE = 8T+18                       MP00206Ø
C    MPCDM     CONVERTS DOUBLE-PRECISION TO MULTIPLE-PRECISION           MP00207Ø
C    MPCHK     PRINTS ERROR MESSAGE ON UNIT LUN IF B, T, M OR MXR        MP00208Ø
C              IS ILLEGAL, OR ON UNIT 6 IF LUN IS ILLEGAL                MP00209Ø
C              (LUN SHOULD BE IN RANGE 1 TO 99)                          MP00210Ø
C *  MPCIM     CONVERTS INTEGER TO MULTIPLE-PRECISION                    MP00211Ø
C              CALL MPCIM (IX, Z) MEANS Z = IX                           MP00212Ø
C    MPCLR     SETS SOME DIGITS OF AN MP NUMBER TO ZERO                  MP00213Ø
C    MPCMD     CONVERTS AN MP NUMBER TO DOUBLE-PRECISION REAL            MP00214Ø
C    MPCMDE    CONVERTS AN MP NUMBER TO (DOUBLE-PRECISION)               MP00215Ø
C              FRACTION AND (DECIMAL) EXPONENT,                          MP00216Ø
C              SPACE = 6T+14                                             MP00217Ø
C    MPCMEF    CONVERTS MP NUMBER TO FRACTION AND (DECIMAL)              MP00218Ø
C              EXPONENT, SPACE = 5T+12                                   MP00219Ø
C *  MPCMF     FINDS FRACTIONAL PART OF AN MP NUMBER                     MP00220Ø
C    MPCMI     CONVERTS AN MP NUMBER TO AN INTEGER                       MP00221Ø
C *  MPCMIM    CONVERTS AN MP NUMBER TO A MULTIPLE-PRECISION INTEGER     MP00222Ø
C *  MPCMPA    COMPARES ABSOLUTE VALUES OF TWO MP NUMBERS                MP00223Ø
C              MPCMPA (X, Y) RETURNS SIGN(ABS(X)-ABS(Y))                 MP00224Ø
C *  MPCMPI    COMPARES AN MP NUMBER WITH AN INTEGER, SPACE = 2T+6       MP00225Ø
C              MPCMPI (X, I) RETURNS SIGN(X-I)                           MP00226Ø
C *  MPCMPR    COMPARES AN MP NUMBER WITH A REAL, SPACE = 2T+6           MP00227Ø
C              MPCMPR (X, RI) RETURNS SIGN(X-RI)                         MP00228Ø
C *  MPCMR     CONVERTS AN MP NUMBER TO (SINGLE-PRECISION) REAL          MP00229Ø
C              CALL MPCMR (X, RZ) MEANS RZ = SNGL(X)                     MP00230Ø
C *  MPCMRE    CONVERTS AN MP NUMBER TO EXPONENT AND                     MP00231Ø
C              (SINGLE-PRECISION) FRACTION, I.E. F*1Ø**I                 MP00232Ø
C              SPACE = 6T+14                                             MP00233Ø
C *  MPCOMP    COMPARES TWO MP NUMBERS                                   MP00234Ø
C              MPCOMP (X, Y) RETURNS SIGN(X-Y)                           MP00235Ø
C *  MPCOS     COMPUTES COSINE OF AN MP NUMBER, USING AN                 MP00236Ø
C              O(M(T)T/LOG(T)) METHOD, SPACE = 5T+12                     MP00237Ø
```

```
C                CALL MPCOS (X, Y) MEANS Y = COS(X)                          MP00238Ø
C *  MPCOSH      COMPUTES. HYPERBOLIC COSINE OF AN MP NUMBER                 MP00239Ø
C                USING MPEXP, SPACE = 5T+12                                  MP00240Ø
C                CALL MPCOSH (X, Y) MEANS Y = COSH(X)                        MP00241Ø
C *  MPCQM       CONVERTS A RATIONAL NUMBER TO MULTIPLE-PRECISION            MP00242Ø
C                CALL MPCQM (I, J, Q) MEANS Q = I/J                          MP00243Ø
C    MPCRM       CONVERTS REAL TO MULTIPLE-PRECISION                         MP00244Ø
C                CALL MPCRM (RX, Z) MEANS Z = RX                             MP00245Ø
C *  MPDAW       COMPUTES DAWSONS INTEGRAL, DAW(X) = EXP(-X**2)*(INTEGRAL    MP00246Ø
C                FROM Ø TO X OF EXP(U**2) DU),  SPACE = 5T+17                MP00247Ø
C                CALL MPDAW (X, Y) MEANS Y = DAW(X)                          MP00248Ø
C *  MPDIV       DIVIDES TWO MP NUMBERS, SPACE = 4T+1Ø                       MP00249Ø
C                CALL MPDIV (X, Y, Z) MEANS Z = X/Y                          MP00250Ø
C *  MPDIVI      DIVIDES AN MP NUMBER BY AN INTEGER                          MP00251Ø
C                USING AN O(T) METHOD (MUCH FASTER THAN MPDIV)               MP00252Ø
C                CALL MPDIVI (X, IY, Z) MEANS Z = X/IY                       MP00253Ø
C    MPDUMP      DUMPS AN MP NUMBER (USEFUL FOR DEBUGGING)                   MP00254Ø
C                CALL MPDUMP (X) DUMPS THE MP NUMBER X ON UNIT LUN           MP00255Ø
C *  MPEI        EVALUATES EXPONENTIAL INTEGRAL OF AN MP NUMBER,             MP00256Ø
C                SPACE = 19T+31                                              MP00257Ø
C                CALL MPEI (X, Y) MEANS Y = EI(X)                            MP00258Ø
C *  MPEPS       COMPUTES THE (MULTIPLE-PRECISION) MACHINE PRECISION         MP00259Ø
C                CALL MPEPS (X) MEANS X = Ø.5*B**(1-T) IF B EVEN             MP00260Ø
C *  MPERF       COMPUTES ERROR FUNCTION OF AN MP NUMBER,                    MP00261Ø
C                SPACE = 5T+12                                               MP00262Ø
C                CALL MPERF (X, Y) MEANS Y = ERF(X)                          MP00263Ø
C *  MPERFC      COMPUTES COMPLEMENTARY ERROR FUNCTION OF AN MP NUMBER,      MP00264Ø
C                SPACE = 12T+26                                              MP00265Ø
C                CALL MPERFC (X, Y)  MEANS Y = ERFC(X)                       MP00266Ø
C    MPERF2      COMPUTES EXP(X*X)*(INTEGRAL FROM Ø TO X OF                  MP00267Ø
C                EXP(-U*U) DU), CALLED BY MPERF, SPACE = 5T+12               MP00268Ø
C    MPERF3      ROUTINE CALLED BY MPERF, MPDAW AND MPERFC,                  MP00269Ø
C                SPACE = 4T+1Ø                                               MP00270Ø
C    MPERR       ERROR HANDLING ROUTINE (TERMINATES EXECUTION AT             MP00271Ø
C                PRESENT BUT MAY EASILY BE MODIFIED).                        MP00272Ø
C *  MPEUL       RETURNS EULERS CONSTANT (GAMMA = Ø.57721566...) TO          MP00273Ø
C                MULTIPLE-PRECISION ACCURACY, SPACE = 5T+14                  MP00274Ø
C                CALL MPEUL (G) MEANS G = Ø.57721566...                      MP00275Ø
C *  MPEXP       COMPUTES EXPONENTIAL OF A MULTIPLE-PRECISION                MP00276Ø
C                NUMBER, USING AN O(SQRT(T)M(T)) METHOD, SPACE = 4T+1Ø       MP00277Ø
C                CALL MPEXP (X, Y) MEANS Y = EXP(X)                          MP00278Ø
C    MPEXP1      COMPUTES EXP(X)-1 FOR ABS(X) .LT. 1  (CALLED BY             MP00279Ø
C                MPEXP, MPSINH AND MPTANH),  SPACE = 3T+8                    MP00280Ø
C    MPEXT       A ROUNDING ROUTINE CALLED BY MPDIV AND MPSQRT               MP00281Ø
C *  MPGAM       COMPUTES GAMMA FUNCTION OF AN MP ARGUMENT,                  MP00282Ø
C                SPACE SAME AS FOR MPLNGM (IN WORST CASE)                    MP00283Ø
C                CALL MPGAM (X, Y) MEANS Y = GAMMA(X)                        MP00284Ø
C    MPGAMQ      COMPUTES GAMMA FUNCTION OF A RATIONAL ARGUMENT,             MP00285Ø
C                USING AN O(T**2) METHOD, SPACE = 6T+12                      MP00286Ø
C                CALL MPGAMQ (I, J, X) MEANS X = GAMMA(I/J)                  MP00287Ø
C    MPGCD       DIVIDES TWO INTEGERS BY THEIR GREATEST COMMON DIVISOR       MP00288Ø
C                (CALLED BY MPMULQ, MPGAMQ, ETC)                             MP00289Ø
C    MPHANK      ROUTINE CALLED BY MPBESJ (EVALUATES HANKELS ASYMPTOTIC      MP00290Ø
C                SERIES FOR BESSEL FUNCTIONS), SPACE = 11T+24                MP00291Ø
C *  MPIN        CONVERTS FIXED-POINT NUMBER READ UNDER A1 FORMAT            MP00292Ø
C                TO MULTIPLE-PRECISION, SPACE = 3T+11                        MP00293Ø
C *  MPINE       SAME AS MPIN BUT RESULT IS MULTIPLIED BY A POWER OF         MP00294Ø
C                TEN (USEFUL FOR READING IN FLOATING-POINT NUMBERS),         MP00295Ø
C                SPACE = 5T+12                                               MP00296Ø
C *  MPLI        EVALUATES LOGARITHMIC INTEGRAL LI(X), SPACE = 19T+31        MP00297Ø
C                CALL MPLI (X, Y) MEANS Y = LI(X)                            MP00298Ø
C *  MPLN        COMPUTES NATURAL LOG OF AN MP NUMBER,                       MP00299Ø
C                USING AN O(SQRT(T).M(T)) METHOD, SPACE = 6T+14              MP00300Ø
C                CALL MPLN (X, Y) MEANS Y = LN(X)                            MP00301Ø
C    MPLNGM      COMPUTES LN(GAMMA(X)) FOR POSITIVE MP X,                    MP00302Ø
C                USING STIRLINGS APPROXIMATION,                              MP00303Ø
C                SPACE = 11T+24+NL*((T+3)/2), WHERE NL IS THE NUMBER         MP00304Ø
C                OF TERMS USED IN THE ASYMPTOTIC EXPANSION,                  MP00305Ø
C                NL .LE. (2 + T*LN(B)/8)                                     MP00306Ø
C                CALL MPLNGM (X, Y) MEANS Y = LN(GAMMA(X))                   MP00307Ø
C    MPLNGS      COMPUTES NATURAL LOG OF AN MP NUMBER, USING                 MP00308Ø
C                THE GAUSS-SALAMIN ALGORITHM.  RECOMMENDED FOR               MP00309Ø
C                TESTING MPLN AND MPLNI ONLY (UNLESS T LARGE).               MP00310Ø
C                SPACE = 6T+26                                               MP00311Ø
C *  MPLNI       COMPUTES NATURAL LOG OF AN INTEGER, USING AN                MP00312Ø
C                O(T**2) METHOD (FASTER THAN MPLN), SPACE = 3T+8             MP00313Ø
```

```
C            CALL MPLNI (N, X) MEANS X = LN(N)                          MPØØ314Ø
C    MPLNS   COMPUTES LN(1+X) FOR SMALL MP X. SPACE = 5T+12             MPØØ315Ø
C    MPL235  COMPUTES NATURAL LOG OF AN INTEGER WHOSE PRIME             MPØØ316Ø
C            FACTORS ARE 2, 3 AND/OR 5 (CALLED BY MPLNI),               MPØØ317Ø
C            SPACE = 3T+8                                               MPØØ318Ø
C *  MPMAX   COMPUTES THE MAXIMUM OF TWO MP NUMBERS                     MPØØ319Ø
C            CALL MPMAX (X, Y, Z) MEANS Z = MAX(X,Y)                    MPØØ32ØØ
C *  MPMAXR  COMPUTES THE LARGEST POSITIVE MP NUMBER                    MPØØ321Ø
C            CALL MPMAXR (X) MEANS X = MP NUMBER WITH EXPONENT M        MPØØ322Ø
C            AND ALL DIGITS B-1                                         MPØØ323Ø
C *  MPMIN   COMPUTES THE MINIMUM OF TWO MP NUMBERS                     MPØØ324Ø
C            CALL MPMIN (X, Y, Z) MEANS Z = MIN(X,Y)                    MPØØ325Ø
C *  MPMINR  RETURNS THE SMALLEST NORMALIZED POSITIVE MP NUMBER         MPØØ326Ø
C            CALL MPMINR (X) MEANS X = B**(-M-1)                        MPØØ327Ø
C    MPMLP   INNER LOOP ROUTINE CALLED BY MPMUL                         MPØØ328Ø
C *  MPMUL   MULTIPLIES TWO MP NUMBERS                                  MPØØ329Ø
C            USING AN M(T) = O(T**2) ALGORITHM                          MPØØ33ØØ
C            CALL MPMUL (X, Y, Z) MEANS Z = X*Y                         MPØØ331Ø
C *  MPMULI  MULTIPLIES AN MP NUMBER BY AN                              MPØØ332Ø
C            INTEGER USING AN O(T) METHOD (FASTER THAN MPMUL)           MPØØ333Ø
C            CALL MPMULI (X, IY, Z) MEANS Z = X*IY                      MPØØ334Ø
C *  MPMULQ  MULTIPLIES MP NUMBER BY A RATIONAL NUMBER                  MPØØ335Ø
C            CALL MPMULQ (X, I, J, Y) MEANS Y = X*I/J                   MPØØ336Ø
C    MPMUL2  ROUTINE CALLED BY MPMULI                                   MPØØ337Ø
C *  MPNEG   REVERSES SIGN OF AN MP NUMBER                              MPØØ338Ø
C            CALL MPNEG (X, Y) MEANS Y = -X                             MPØØ339Ø
C    MPNZR   NORMALIZES AND ROUNDS OR TRUNCATES (CALLED BY              MPØØ34ØØ
C            MPADD2, MPDIVI, MPMUL AND MPMUL2)                          MPØØ341Ø
C *  MPOUT   CONVERTS MULTIPLE-PRECISION TO A FORM SUITABLE FOR          MPØØ342Ø
C            PRINTING UNDER A1 FORMAT (CORRESPONDS TO F OR I            MPØØ343Ø
C            FORMATS), SPACE = 3T+11                                    MPØØ344Ø
C *  MPOUTE  SIMILAR TO MPOUT BUT GIVES (DECIMAL) EXPONENT AND          MPØØ345Ø
C            FRACTION (CORRESPONDS TO E FORMAT), SPACE = 6T+14          MPØØ346Ø
C    MPOUT2  SAME AS MPOUT BUT ANY BASE FROM 2 TO 16 MAY BE             MPØØ347Ø
C            USED FOR OUTPUT REPRESENTATION, SPACE = 3T+11              MPØØ348Ø
C    MPOVFL  ROUTINE CALLED ON MULTIPLE-PRECISION OVERFLOW              MPØØ349Ø
C            (CALLS MPERR AT PRESENT BUT EASILY MODIFIED)               MPØØ35ØØ
C *  MPPACK  PACKS MP NUMBERS INTO ARRAYS OF DIMENSION                  MPØØ351Ø
C            (T+3)/2  (USEFUL TO SAVE SPACE),                           MPØØ352Ø
C            UNPACKING MAY BE PERFORMED WITH MPUNPK                     MPØØ353Ø
C *  MPPI    RETURNS PI TO MULTIPLE-PRECISION ACCURACY,                 MPØØ354Ø
C            USING AN O(T**2) METHOD, SPACE = 3T+8                      MPØØ355Ø
C            CALL MPPI (X) MEANS X = 3.14159265...                      MPØØ356Ø
C    MPPIGL  RETURNS PI TO MULTIPLE-PRECISION ACCURACY,                 MPØØ357Ø
C            USING GAUSS-LEGENDRE O(LOG(T)M(T)) METHOD,                 MPØØ358Ø
C            RECOMMENDED FOR TESTING MPPI ONLY, SPACE = 6T+14           MPØØ359Ø
C *  MPPOLY  EVALUATES A POLYNOMIAL WITH INTEGER COEFFICIENTS,          MPØØ36ØØ
C            SPACE = 3T+8                                               MPØØ361Ø
C *  MPPWR   RAISES MP NUMBER TO INTEGER POWER,                         MPØØ362Ø
C            SPACE = 4T+1Ø                                              MPØØ363Ø
C            CALL MPPWR (X, N, Y) MEANS Y = X**N                        MPØØ364Ø
C *  MPPWR2  RAISES NONNEGATIVE MP NUMBER TO MP POWER,                  MPØØ365Ø
C            SPACE = 7T+16                                              MPØØ366Ø
C            CALL MPPWR2 (X, Y, Z) MEANS Z = X**Y                       MPØØ367Ø
C *  MPQPWR  RAISES RATIONAL NUMBER TO RATIONAL POWER,                  MPØØ368Ø
C            SPACE = 4T+1Ø                                              MPØØ369Ø
C            CALL MPQPWR (I, J, K, L, X) MEANS X = (I/J)**(K/L)         MPØØ37ØØ
C *  MPREC   FORMS RECIPROCAL OF MP NUMBER,                             MPØØ371Ø
C            USING NEWTONS METHOD, SPACE = 4T+1Ø                        MPØØ372Ø
C            CALL MPREC (X, Y) MEANS Y = 1/X                            MPØØ373Ø
C *  MPROOT  COMPUTES THE N-TH ROOT OF AN MP NUMBER                     MPØØ374Ø
C            USING NEWTONS METHOD, SPACE = 4T+1Ø                        MPØØ375Ø
C            CALL MPROOT (X, N, Y) MEANS Y = X**(1/N)                   MPØØ376Ø
C *  MPSET   SETS THE BASE B AND DIGITS T ETC GIVEN THE                 MPØØ377Ø
C            EQUIVALENT NUMBER OF DECIMAL PLACES REQUIRED               MPØØ378Ø
C            WARNING - MAY CAUSE AN INTEGER OVERFLOW,                   MPØØ379Ø
C            *******   FOR DETAILS SEE COMMENTS IN MPSET                MPØØ38ØØ
C *  MPSIN   COMPUTES SINE OF AN MP NUMBER,                             MPØØ381Ø
C            USING AN O(M(T)T/LOG(T)) METHOD, SPACE = 5T+12             MPØØ382Ø
C            CALL MPSIN (X, Y) MEANS Y = SIN(X)                         MPØØ383Ø
C *  MPSINH  COMPUTES HYPERBOLIC SINE OF AN MP NUMBER,                  MPØØ384Ø
C            USING MPEXP, SPACE = 5T+12                                 MPØØ385Ø
C            CALL MPSINH (X, Y) MEANS Y = SINH(X)                       MPØØ386Ø
C    MPSIN1  COMPUTES SIN(X) OR COS(X) FOR ABS(X) .LE. 1, CALLED        MPØØ387Ø
C            BY MPSIN, MPCOS AND MPTAN, SPACE = 3T+8                    MPØØ388Ø
C *  MPSQRT  COMPUTES SQUARE ROOT OF A NONNEGATIVE MP NUMBER,           MPØØ389Ø
```

```
C                USING NEWTONS METHOD, SPACE = 4T+1Ø                    MPØØ39ØØ
C                CALL MPSQRT (X, Y) MEANS Y = SQRT(X)                   MPØØ391Ø
C *  MPSTR       STORES ONE MP NUMBER IN ANOTHER                        MPØØ392Ø
C                CALL MPSTR (X, Y) MEANS Y = X                          MPØØ393Ø
C *  MPSUB       SUBTRACTS ONE MP NUMBER FROM ANOTHER                   MPØØ394Ø
C                CALL MPSUB (X, Y, Z) MEANS Z = X - Y                   MPØØ395Ø
C *  MPTAN       COMPUTES TAN OF AN MP NUMBER,                          MPØØ396Ø
C                USING MPSIN1, SPACE = 6T+2Ø                            MPØØ397Ø
C                CALL MPTAN (X, Y) MEANS Y = TAN(X)                     MPØØ398Ø
C *  MPTANH      COMPUTES HYPERBOLIC TAN OF AN MP NUMBER,               MPØØ399Ø
C                USING MPEXP, SPACE = 5T+12                             MPØØ4ØØØ
C                CALL MPTANH (X, Y) MEANS Y = TANH(X)                   MPØØ4Ø1Ø
C    MPUNFL      ROUTINE CALLED ON MULTIPLE-PRECISION UNDERFLOW         MPØØ4Ø2Ø
C                (SETS RESULT TO ZERO AT PRESENT BUT EASILY MODIFIED)   MPØØ4Ø3Ø
C *  MPUNPK      UNPACKS AN ARRAY FORMED BY MPPACK TO GIVE AN MP        MPØØ4Ø4Ø
C                NUMBER IN STANDARD FORMAT                              MPØØ4Ø5Ø
C *  MPZETA      COMPUTES RIEMANN ZETA FUNCTION FOR POSITIVE            MPØØ4Ø6Ø
C                INTEGER ARGUMENTS                                      MPØØ4Ø7Ø
C                SPACE = 8T+18+NL*((T+3)/2), WHERE NL IS THE            MPØØ4Ø8Ø
C                NUMBER OF TERMS USED IN THE ASYMPTOTIC                 MPØØ4Ø9Ø
C                EXPANSION, NL .LE. (1 + Ø.1*T*LN(B))                   MPØØ41ØØ
C                CALL MPZETA (N, X) MEANS X = ZETA(N)                   MPØØ411Ø
C *  MP4ØD       OUTPUT ROUTINE CALLED BY TEST PROGRAM,                 MPØØ412Ø
C                USEFUL FOR EASY FIXED-POINT OUTPUT,                    MPØØ413Ø
C                SPACE = 3T+N+14 FOR N DECIMAL PLACE OUTPUT             MPØØ414Ø
C                CALL MP4ØD (N, X) WRITES X TO N DECIMAL PLACES ON UNIT MPØØ415Ø
C                LUN, ASSUMING ABS(X) .LT. 1Ø                           MPØØ416Ø
C    MP4ØE       OUTPUT ROUTINE CALLED BY MP4ØD                         MPØØ417Ø
C *  MP4ØF       OUTPUT ROUTINE CALLED BY TEST2 PROGRAM,                MPØØ418Ø
C                USEFUL FOR EASY FLOATING-POINT OUTPUT,                 MPØØ419Ø
C                SPACE = 6T+N+17 FOR N SIGNIFICANT FIGURE OUTPUT        MPØØ42ØØ
C                CALL MP4ØF (N, X) WRITES X TO N SIGNIFICANT FIGURES    MPØØ421Ø
C                (IN DECIMAL EXPONENT AND FRACTION FORM) ON UNIT LUN    MPØØ422Ø
C    MP4ØG       OUTPUT ROUTINE CALLED BY MP4ØF                         MPØØ423Ø
C *  TEST        A MAIN PROGRAM WHICH TESTS SOME OF THE ROUTINES IN MP  MPØØ424Ø
C                WHILE COMPUTING VARIOUS CONSTANTS TO 4Ø DECIMAL PLACES MPØØ425Ø
C    TESTV       A VERSION OF TEST WITH VARIABLE-PRECISION COMPUTATION  MPØØ426Ø
C                AND OUTPUT                                             MPØØ427Ø
C *  TEST2       ANOTHER TEST PROGRAM WHICH TESTS ROUTINES              MPØØ428Ø
C                NOT CALLED BY TEST OR TESTV                            MPØØ429Ø
C    TIMEMP      A MACHINE-DEPENDENT FUNCTION CALLED BY TESTV, SHOULD   MPØØ43ØØ
C                BE MODIFIED BY THE USER BEFORE TESTV IS RUN.           MPØØ431Ø
C                                                                       MPØØ432Ø
C INDEX                                                                 MPØØ433Ø
C *****                                                                 MPØØ434Ø
C                                                                       MPØØ435Ø
C THE STARTING LINE SEQUENCE NUMBERS (GIVEN IN COLUMNS 73-8Ø)           MPØØ436Ø
C OF THE MP ROUTINES ARE AS FOLLOWS.                                    MPØØ437Ø
C                                                                       MPØØ438Ø
C     COMMENTS     MPØØØØ3Ø                                           MPØØ439Ø
C     EXAMPLE      MPØØ546Ø                                           MPØØ44ØØ
C     MPABS        MPØØ619Ø                                           MPØØ441Ø
C     MPADD        MPØØ627Ø                                           MPØØ442Ø
C     MPADDI       MPØØ635Ø                                           MPØØ443Ø
C     MPADDQ       MPØØ651Ø                                           MPØØ444Ø
C     MPADD2       MPØØ663Ø                                           MPØØ445Ø
C     MPADD3       MPØØ723Ø                                           MPØØ446Ø
C     MPART1       MPØØ813Ø                                           MPØØ447Ø
C     MPASIN       MPØØ861Ø                                           MPØØ448Ø
C     MPATAN       MPØØ899Ø                                           MPØØ449Ø
C     MPBERN       MPØØ961Ø                                           MPØØ45ØØ
C     MPBESJ       MPØ1Ø76Ø                                           MPØØ451Ø
C     MPBES2       MPØ119ØØ                                           MPØØ452Ø
C     MPCDM        MPØ1258Ø                                           MPØØ453Ø
C     MPCHK        MPØ1325Ø                                           MPØØ454Ø
C     MPCIM        MPØ1376Ø                                           MPØØ455Ø
C     MPCLR        MPØ143Ø                                            MPØØ456Ø
C     MPCMD        MPØ1416Ø                                           MPØØ457Ø
C     MPCMDE       MPØ1456Ø                                           MPØØ458Ø
C     MPCMEF       MPØ1479Ø                                           MPØØ459Ø
C     MPCMF        MPØ155ØØ                                           MPØØ46ØØ
C     MPCMI        MPØ1582Ø                                           MPØØ461Ø
C     MPCMIM       MPØ1625Ø                                           MPØØ462Ø
C     MPCMPA       MPØ1648Ø                                           MPØØ463Ø
C     MPCMPI       MPØ1664Ø                                           MPØØ464Ø
C     MPCMPR       MPØ168ØØ                                           MPØØ465Ø
```

C	MPCMR	MPØ1696Ø	MPØØ466Ø
C	MPCMRE	MPØ1731Ø	MPØØ467Ø
C	MPCOMP	MPØ1753Ø	MPØØ468Ø
C	MPCOS	MPØ1788Ø	MPØØ469Ø
C	MPCOSH	MPØ1817Ø	MPØØ47ØØ
C	MPCQM	MPØ1843Ø	MPØØ471Ø
C	MPCRM	MPØ1863Ø	MPØØ472Ø
C	MPDAW	MPØ1926Ø	MPØØ473Ø
C	MPDIV	MPØ199ØØ	MPØØ474Ø
C	MPDIVI	MPØ2Ø38Ø	MPØØ475Ø
C	MPDUMP	MPØ2151Ø	MPØØ476Ø
C	MPEI	MPØ2177Ø	MPØØ477Ø
C	MPEPS	MPØ2295Ø	MPØØ478Ø
C	MPERF	MPØ2324Ø	MPØØ479Ø
C	MPERFC	MPØ2381Ø	MPØØ48ØØ
C	MPERF2	MPØ2431Ø	MPØØ481Ø
C	MPERF3	MPØ2484Ø	MPØØ482Ø
C	MPERR	MPØ2539Ø	MPØØ483Ø
C	MPEUL	MPØ2558Ø	MPØØ484Ø
C	MPEXP	MPØ2634Ø	MPØØ485Ø
C	MPEXP1	MPØ2737Ø	MPØØ486Ø
C	MPEXT	MPØ28ØØØ	MPØØ487Ø
C	MPGAM	MPØ2824Ø	MPØØ488Ø
C	MPGAMQ	MPØ29Ø5Ø	MPØØ489Ø
C	MPGCD	MPØ3Ø3ØØ	MPØØ49ØØ
C	MPHANK	MPØ3Ø56Ø	MPØØ491Ø
C	MPIN	MPØ3139Ø	MPØØ492Ø
C	MPINE	MPØ3257Ø	MPØØ493Ø
C	MPLI	MPØ33Ø2Ø	MPØØ494Ø
C	MPLN	MPØ3338Ø	MPØØ495Ø
C	MPLNGM	MPØ3394Ø	MPØØ496Ø
C	MPLNGS	MPØ3481Ø	MPØØ497Ø
C	MPLNI	MPØ3558Ø	MPØØ498Ø
C	MPLNS	MPØ3663Ø	MPØØ499Ø
C	MPL235	MPØ373ØØ	MPØØ5ØØØ
C	MPMAX	MPØ3783Ø	MPØØ5Ø1Ø
C	MPMAXR	MPØ3795Ø	MPØØ5Ø2Ø
C	MPMIN	MPØ3811Ø	MPØØ5Ø3Ø
C	MPMINR	MPØ3823Ø	MPØØ5Ø4Ø
C	MPMLP	MPØ3839Ø	MPØØ5Ø5Ø
C	MPMUL	MPØ3849Ø	MPØØ5Ø6Ø
C	MPMULI	MPØ3924Ø	MPØØ5Ø7Ø
C	MPMULQ	MPØ3934Ø	MPØØ5Ø8Ø
C	MPMUL2	MPØ396ØØ	MPØØ5Ø9Ø
C	MPNEG	MPØ4Ø48Ø	MPØØ51ØØ
C	MPNZR	MPØ4Ø56Ø	MPØØ511Ø
C	MPOUT	MPØ4137Ø	MPØØ512Ø
C	MPOUTE	MPØ4152Ø	MPØØ513Ø
C	MPOUT2	MPØ4184Ø	MPØØ514Ø
C	MPOVFL	MPØ4323Ø	MPØØ515Ø
C	MPPACK	MPØ4344Ø	MPØØ516Ø
C	MPPI	MPØ4371Ø	MPØØ517Ø
C	MPPIGL	MPØ4397Ø	MPØØ518Ø
C	MPPOLY	MPØ4442Ø	MPØØ519Ø
C	MPPWR	MPØ447ØØ	MPØØ52ØØ
C	MPPWR2	MPØ4513Ø	MPØØ521Ø
C	MPQPWR	MPØ4544Ø	MPØØ522Ø
C	MPREC	MPØ4617Ø	MPØØ523Ø
C	MPROOT	MPØ4698Ø	MPØØ524Ø
C	MPSET	MPØ48Ø4Ø	MPØØ525Ø
C	MPSIN	MPØ4881Ø	MPØØ526Ø
C	MPSINH	MPØ4948Ø	MPØØ527Ø
C	MPSIN1	MPØ4986Ø	MPØØ528Ø
C	MPSQRT	MPØ5Ø44Ø	MPØØ529Ø
C	MPSTR	MPØ5Ø68Ø	MPØØ53ØØ
C	MPSUB	MPØ5Ø89Ø	MPØØ531Ø
C	MPTAN	MPØ5Ø98Ø	MPØØ532Ø
C	MPTANH	MPØ5161Ø	MPØØ533Ø
C	MPUNFL	MPØ52ØØØ	MPØØ534Ø
C	MPUNPK	MPØ5215Ø	MPØØ535Ø
C	MPZETA	MPØ5244Ø	MPØØ536Ø
C	MP4ØD	MPØ5359Ø	MPØØ537Ø
C	MP4ØE	MPØ5376Ø	MPØØ538Ø
C	MP4ØF	MPØ539ØØ	MPØØ539Ø
C	MP4ØG	MPØ54Ø8Ø	MPØØ54ØØ
C	TEST	MPØ5422Ø	MPØØ541Ø

```
C     TESTV        MPØ56Ø3Ø                                                MPØØ542Ø
C     TEST2        MPØ57Ø4Ø                                                MPØØ543Ø
C     TIMEMP       MPØ6414Ø                                                MPØØ544Ø
C                                                                           MPØØ545Ø
C $$                  ******  EXAMPLE  ******                              MPØØ546Ø
C                                                                           MPØØ547Ø
C THIS PROGRAM COMPUTES PI AND EXP(PI*SQRT(163/9)) TO 1ØØ                  MPØØ548Ø
C DECIMAL PLACES, AND EXP(PI*SQRT(163)) TO 9Ø DECIMAL PLACES,              MPØØ549Ø
C AND WRITES THEM ON LOGICAL UNIT 6.  EXECUTION                            MPØØ55ØØ
C TIME ON A UNIVAC 11Ø8 (WITH FORTRAN SE1D) IS 1.Ø51 SECONDS.              MPØØ551Ø
C                                                                           MPØØ552Ø
C TO RUN EXAMPLE THE FOLLOWING MP ROUTINES ARE REQUIRED - MPABS,           MPØØ553Ø
C MPADD, MPADDI, MPADD2, MPADD3, MPART1, MPCHK, MPCIM, MPCLR, MPCMF,       MPØØ554Ø
C MPCMI, MPCMPR, MPCMR, MPCOMP, MPCQM, MPCRM, MPDIVI, MPERR,               MPØØ555Ø
C MPEXP, MPEXP1, MPGCD, MPLNI, MPL235, MPMAXR, MPMLP, MPMUL,               MPØØ556Ø
C MPMULI, MPMULQ, MPMUL2, MPNZR, MPOUT, MPOUT2, MPOVFL, MPPI,              MPØØ557Ø
C MPPWR, MPQPWR, MPREC, MPROOT, MPSET, MPSTR, MPSUB, MPUNFL.               MPØØ558Ø
C                                                                           MPØØ559Ø
C CORRECT OUTPUT (EXCLUDING HEADINGS) IS AS FOLLOWS                        MPØØ56ØØ
C                                                                           MPØØ561Ø
C                 3.14159265358979323846264338327950288419716939937510 MPØØ562Ø
C                   582Ø974944592307816406286208998628Ø348253421170680 MPØØ563Ø
C             64Ø32Ø.ØØØØØØØØØ6Ø4863735Ø49Ø16Ø3947174181881853947577148 MPØØ564Ø
C                   576Ø3665918194652218258286942536340815822646477590 MPØØ565Ø
C 262537412640768743.99999999999925ØØ725971981856888793538563373369908 6 MPØØ566Ø
C                   27Ø753741Ø37821Ø64791Ø1186Ø7312951181346               MPØØ567Ø
C                                                                           MPØØ568Ø
C CERTAIN PARAMETERS AND WORKING SPACE IN COMMON.                          MPØØ569Ø
      COMMON B, T, M, LUN, MXR, R                                          MPØØ57ØØ
C                                                                           MPØØ571Ø
C MPEXP REQUIRES 4T+1Ø WORDS AND WE HAVE T .LE. 62 IF WORDLENGTH           MPØØ572Ø
C AT LEAST 16 BITS, SO 4T+1Ø .LE. 258.  DIMENSIONS CAN BE REDUCED          MPØØ573Ø
C IF WORDLENGTH IS GREATER THAN 16 BITS.                                   MPØØ574Ø
      INTEGER B, T, R(258)                                                 MPØØ575Ø
C                                                                           MPØØ576Ø
C VARIABLES NEED T+2 .LE. 64 WORDS AND ALLOW 11Ø CHARACTERS FOR            MPØØ577Ø
C DECIMAL OUTPUT                                                           MPØØ578Ø
      INTEGER PI(64), X(64), C(11Ø)                                        MPØØ579Ø
C                                                                           MPØØ58ØØ
C CALL MPSET TO SET OUTPUT LOGICAL UNIT = 6 AND EQUIVALENT                 MPØØ581Ø
C NUMBER OF DECIMAL PLACES TO AT LEAST 11Ø.  THE LAST TWO                  MPØØ582Ø
C PARAMETERS ARE THE DIMENSIONS OF PI (OR X) AND R.                        MPØØ583Ø
      CALL MPSET (6, 11Ø, 64, 258)                                         MPØØ584Ø
C                                                                           MPØØ585Ø
C COMPUTE MULTIPLE-PRECISION PI                                            MPØØ586Ø
      CALL MPPI(PI)                                                        MPØØ587Ø
C                                                                           MPØØ588Ø
C CONVERT TO PRINTABLE FORMAT (F11Ø.1ØØ) AND WRITE                         MPØØ589Ø
      CALL MPOUT (PI, C, 11Ø, 1ØØ)                                        MPØØ59ØØ
      WRITE (LUN, 1Ø) B, T, C                                              MPØØ591Ø
   1Ø FORMAT (32H1EXAMPLE OF MP PACKAGE,   BASE =, I9,                     MPØØ592Ø
     $  12H,   DIGITS =, I4 /// 11H PI TO 1ØØD //                          MPØØ593Ø
     $  11X, 6ØA1 / 21X, 5ØA1)                                            MPØØ594Ø
C                                                                           MPØØ595Ø
C SET X = SQRT(163/9), THEN MULTIPLY BY PI                                 MPØØ596Ø
      CALL MPQPWR (163, 9, 1, 2, X)                                        MPØØ597Ø
      CALL MPMUL (X, PI, X)                                                MPØØ598Ø
C                                                                           MPØØ599Ø
C SET X = EXP(X)                                                           MPØØ6ØØØ
      CALL MPEXP (X, X)                                                    MPØØ6Ø1Ø
C                                                                           MPØØ6Ø2Ø
C CONVERT TO PRINTABLE FORMAT AND WRITE                                    MPØØ6Ø3Ø
      CALL MPOUT (X, C, 11Ø, 1ØØ)                                          MPØØ6Ø4Ø
      WRITE (LUN, 2Ø) C                                                    MPØØ6Ø5Ø
   2Ø FORMAT (/ 28H EXP(PI*SQRT(163/9)) TO 1ØØD //                         MPØØ6Ø6Ø
     $        11X, 6ØA1 / 21X, 5ØA1)                                       MPØØ6Ø7Ø
C                                                                           MPØØ6Ø8Ø
C SET X = X**3 = EXP(PI*SQRT(163))                                         MPØØ6Ø9Ø
      CALL MPPWR (X, 3, X)                                                 MPØØ61ØØ
C                                                                           MPØØ611Ø
C WRITE IN FORMAT F11Ø.9Ø                                                  MPØØ612Ø
      CALL MPOUT (X, C, 11Ø, 9Ø)                                           MPØØ613Ø
      WRITE (LUN, 3Ø) C                                                    MPØØ614Ø
   3Ø FORMAT (/ 25H EXP(PI*SQRT(163)) TO 9ØD //                            MPØØ615Ø
     $        1X, 7ØA1 / 21X, 4ØA1)                                        MPØØ616Ø
      STOP                                                                 MPØØ617Ø
      END                                                                  MPØØ618Ø
```

ACM Transactions on Mathematical Software, Vol. 5, No. 4, December 1979, Pages 518–519.

REMARK ON ALGORITHM 524

MP, A Fortran Multiple-Precision Arithmetic Package [A1]
[R.P. Brent, *ACM Trans. Math. Software 4*, 1 (March 1978), 71–81]

R.P. Brent [Recd 7 Aug. 1978 and 6 Dec. 1978]
Department of Computer Science, Australian National University, P.O. Box 4, Canberra, ACT 2600, Australia

A new version of the Fortran multiple-precision arithmetic package MP, which is described in [1] and given as ACM Algorithm 524, is now available from the ACM Algorithms Distribution Service. The new version may be used with the Augment preprocessor [4], and the necessary interface routines and description deck, described in [2], are supplied. The MP Users' Guide (also supplied with the package) has been revised to describe the Augment interface routines and the use of MP via Augment.

The new version incorporates faster algorithms for the exponential integral and Euler's constant [3]. Consequently, the TEST2 program described in [1] now runs about 10 percent faster.

In versions of MP dated June 7, 1978, and earlier (including Algorithm 524), there may be an error in the least significant digit when multiple-precision numbers with the same sign and exponent are added using subroutine MPADD. To correct this, change .LT. to .LE. in line MP007390 of MPADD3.

ACKNOWLEDGMENT

I am indebted to John P. Jeter of the University of South West Louisiana for finding both the error in MPADD3 and the required correction.

REFERENCES

1. BRENT, R.P. A Fortran multiple-precision arithmetic package. *ACM Trans. Math. Software 4*, 1 (March 1978), 57–70.
2. BRENT, R.P., HOOPER, J.A., AND YOHE, J.M. An Augment interface for Brent's multiple-precision arithmetic package. Mathematics Res. Ctr., U. of Wisconsin, Madison, Wis., Aug. 1978.
3. BRENT, R.P., AND MCMILLAN, E.M. Some new algorithms for high-precision computation of Euler's constant. To appear in *Mathematics of Computation*.
4. CRARY, F.D. A versatile precompiler for nonstandard arithmetics. *ACM Trans. Math. Software 5*, 2(June 1979), 204–217.

ALGORITHM

[Only summary information of the revised version of the algorithm is printed here. This listing is available from the ACM Algorithms Distribution Service (see inside back cover for order form) and will be supplied to those requesting Algorithm 524.]

NAME(n):	indicates a Fortran module from the MP package with n records
NAME^{E}(n):	indicates "NAME" is included as an example
NAME^{T}(n):	indicates "NAME" is part of the test package
NAME^{U}(n):	indicates a listing of the user guide
NAME^{A}(n):	indicates an Augment description deck and Jacobi program using it
Contents:	EXAMPLE^{T}(111), MPABS(8), MPADD(8), MPADDI(16), MPADDQ(12), MPADD2(60), MPADD3(90), MPART1(48), MPASIN(38), MPATAN(62), MPBASA(9), MPBASB(13), MPBERN(115), MPBESJ(114), MPBES2(68), MPCAM(44), MPCDM(67), MPCHK(51), MPCIM(26), MPCLR(13), MPCMD(40), MPCMDE(23), MPCMEF(71), MPCMF(32), MPCMI(43), MPCMIM(23), MPCMPA(16), MPCMPI(16), MPCMPR(16), MPCMR(35), MPCMRE(22), MPCOMP(35), MPCOS(29), MPCOSH(26), MPCQM(20), MPCRM(63),

MPDAW(64), MPDGA(10), MPDGB(21), MPDIGA(9), MPDIGB(14), MPDIV(48), MPDIVI(113), MPDUMP(26), MPEI(156), MPEPS(29), MPEQ(7), MPERF(57), MPERFC(50), MPERF2(53), MPERF3(55), MPERR(19), MPEUL(76), MPEXP(103), MPEXPA(11), MPEXPB(24), MPEXP1(63), MPEXT(24), MPGAM(81), MPGAMQ(125), MPGCD(26), MPGCDA(78), MPGCDB(38), MPGE(7), MPGT(7), MPHANK(83), MPIN(118), MPINE(45), MPINF(27), MPINIT(39), MPIO(19), MPKSTR(15), MPLE(7), MPLI(36), MPLN(56), MPLNGM(87), MPLNGS(77), MPLNI(105), MPLNS(67), MPLT(7), MPL235(53), MPMAX(12), MPMAXR(16), MPMEXA(9), MPMEXB(18), MPMIN(12), MPMINR(16), MPMLP(10), MPMUL(75), MPMULI(10), MPMULQ(26), MPMUL2(88), MPNE(7), MPNEG(8), MPNZR(81), MPOUT(15), MPOUTE(32), MPOUTF(25), MPOUT2(139), MPOVFL(21), MPPACK(27), MPPI(26), MPPIGL(45), MPPOLY(28), MPPWR(43), MPPWR2(31), MPQPWR(73), MPREC(81), MPROOT(102), MPSET(77), MPSIGA(7), MPSIGB(24), MPSIN(67), MPSINH(38), MPSIN1(58), MPSQRT(24), MPSTR(21), MPSUB(9), MPTAN(63), MPTANH(39), MPUNFL(15), MPUNPK(29), MPUPK(45), MPZETA(115), MP40D^{T}(17), MP40E^{T}(14), MP40F^{T}(18), MP40G^{T}(15), TEST^{T}(181), TESTV^{T}(191), TEST2^{T}(620), TIMEMP^{T}(28), GUIDE^{U}(1719), AUGDECK^{A}(159)

525-P 1- 0

ALGORITHM 525
ADAPT, Adaptive Smooth Curve Fitting [E2]

JOHN R. RICE
Purdue University

Key Words and Phrases: piecewise polynomial approximation, adaptive curve fitting, adaptive approximation, Hermite interpolation, function approximation
CR Categories: 5.13
Language: Fortran

DESCRIPTION

1. Introduction and Background

The basic objectives of this function approximation algorithm are: speed, reliability, and smoothness. Algorithms already exist with any two of these three properties. Speed requires that the work be proportional to the length of the curve (for fixed accuracy and more or less uniform complexity of the function). Reliability requires that curves with singularities or near singularities, oscillations, and other complex behavior be handled. Smoothness (number of continuous derivatives) of the approximation obtained is input to the algorithm. The author believes that adaptive piecewise polynomial algorithms offer the best hope for such algorithms with nonadaptive piecewise polynomial schemes or rational function approximation as the only serious competitors. Nonadaptive schemes do not cope efficiently with functions having very nonuniform behavior (singularities in slopes, for example) and the work for rational approximation probably increases faster than linearly with the length of the curve.

The theoretical background for this algorithm is provided by Rice [1] and the references cited there. This may be summarized by saying that for given fixed degree n of the pieces the error of the best piecewise polynomial approximation behaves like k^{-n} where k is the number of pieces. This theoretical result applies to a broad class of functions which includes everything that conceivably could arise in applications. A class of adaptive algorithms has been found which chooses the knots so that the error behaves like k^{-n} even though the best approximation is not obtained. These algorithms apply to any piecewise smooth function with a finite number of "algebraic" singularities, i.e. to any function F(x) which behaves like $a + b(x - s)^{\alpha}$ near the singularity s. The exponent α must give a finite value for the norm of F(x) specified for ADAPT to use, i.e. $\alpha > -\frac{1}{2}$ for least squares, $\alpha > 0$ for uniform approximation.

Received 26 November 1975 and 1 November 1976.

This work was supported in part by the National Science Foundation under Grant GP 32940X.
Author's address: Department of Computer Science, Purdue University, Lafayette, IN 47907.

The key ingredients in these algorithms are a local approximation operator, a local error estimator, and a data structure for the subintervals generated. ADAPT uses Hermite interpolation at subinterval endpoints plus ordinary interpolation in between if needed. Error estimates are made by a simple Gauss quadrature formula and a stack is used for the intervals. This algorithm is probably the simplest choice from those currently known to give the optimal convergence rate. A more detailed description of ADAPT is given in Section 5. Note that the nature of ADAPT *requires* that F(x) and its derivatives be available for arbitrary x (see [2] for guidance on numerically estimating derivatives) and thus ADAPT is not directly applicable to discrete data sets.

Section 2 presents some remarks about the algorithm components; most of them incorporate rather standard methods. Section 3 discusses the use of ADAPT (input/output, role of the arguments CHARF = characteristic length of F(x) and EDIST = error distribution type, and portability). Section 4 describes the extensive testing performed and summarizes the algorithm properties observed in [2]. It also briefly describes the driver program and 20 functions with derivatives used for testing ADAPT which are applicable to similar algorithms. The final section has a very high level description of the algorithm; ADAPT itself is available through the ACM Algorithms Distribution Service.

2. The Principal Algorithm Components

The algorithm ADAPT is to approximate the function F(x) on the interval [A,B] to within an accuracy ACCUR by piecewise polynomials of degree DEGREE with a number SMOOTH of continuous derivatives. The error is measured in the L_p-norm with p set by NORM. Other input is the characteristic length CHARF of F, printed output level LEVEL, the error distribution type EDIST, and the number NBREAK of breakpoints (plus information about the breakpoints if NBREAK > 0). Breakpoints are points where there are breaks in the curve or its derivatives: knots are placed there and special calculations made. The output is the array XKNOTS of knots, the coefficients COEFS (relative to the knot locations) of the polynomial pieces, the estimated error ERROR, and number KNOTS of knots. The approximation obtained is automatically available as the FUNCTION subprogram PPOLY.

2.1 *Data Structure and Discard Procedure for Subintervals.* ADAPT generates a set of subintervals of [A,B] which are maintained in a stack with the leftmost on the top. Since subintervals are created by halving, the maximum size of the stack is limited by the machine word length.

Intervals are discarded whenever the estimated error on an interval is small enough. This decision made in the subprogram CHECK is somewhat subtle and three strategies are provided. This question is discussed later in some detail in connection with the argument EDIST which indicates the strategy selected by the user. Two subprograms, PUT and TAKE, are used for access to the interval stack.

2.2 *Hermite Interpolation.* The values of DEGREE and SMOOTH are specified by the user with DEGREE $\geq 2 *$ SMOOTH $+ 1$. The polynomial pieces are determined by interpolating F and SMOOTH derivatives of F at each endpoint of a subinterval plus the value of F at DEGREE $- 2 *$ SMOOTH $- 1$ other points. During the computation the polynomial pieces are represented by divided differences computed in NEWTON and used by POLYDD. Once a polynomial piece is accepted for the final approximation PTRANS transforms its representation into powers with origin at the left endpoint of the subinterval. The transformation is accomplished by repeated synthetic division. The subprogram COMPUT controls the computation of a polynomial piece as well as the error estimation.

2.3 *The Breakpoint Mechanism.* It is sometimes very useful for a user to be able to specify breaks in some derivative at certain points. Most commonly one has a known or desired jump in the slope at a given point and ADAPT allows this via NBREAK and associated arguments XBREAK, DBREAK, BLEFT, BRIGHT which specify the exact nature of the breakpoint. This rather straightforward facility is implemented primarily in TAKE with some impact on COMPUT.

2.4 *Error Estimation.* The L_P-norm of the local error is estimated in ERRINT by a 4-point Gauss quadrature for (F − POLYDD)**P on the subinterval under consideration. Special code is used for P = infinity (minimax approximation). The global error estimate is built up in PUT by appropriately combining the local error estimates.

2.5 *Fatal Errors.* The algorithm normally terminates when the stack is empty. The stack should not overflow but might do so at very strong singularities which cause the algorithm to want to operate at accuracies inconsistent with the machine word length. This overflow has not occurred in the testing so far, but if it does a message such as the following is printed:

```
INTERVAL DIVIDED TOO MUCH, EXCEEDED LIMIT 50 ON INTERVAL STACK
AT
17923.12345678     17923.12345687
INTERVAL DISCARDED AND COMPUTATION CONTINUED
```

This message may be suppressed by setting LEVEL = −1, but note that the computation is allowed to proceed on the conjecture that this situation is not truly fatal.

A strong singularity has been observed to cause another situation indicated by a message like

```
GOT SHORT INTERVAL **** 3210.12345678   3210.12345679 **** DISCARD IT
```

which may also be suppressed by setting LEVEL = −1. Experiments indicate that the algorithm will recover and produce satisfactory results provided it can discard (and ignore) enough short intervals to get out of the region where the machine word length is inadequate. It often cannot get out of this region before exceeding run time limits or, more likely, the limit on the number of knots. The variable BUFFER in PUT governs short interval detection.

The arrays XKNOTS and COEFS are passed to ADAPT with variable dimensions KDIMEN and NDIMEN. If the number of knots computed exceeds KDIMEN then a fatal error message is printed and the computation aborted. This message cannot be suppressed. Checks are made on various input parameters by SETUP and inconsistent or impossible input leads to fatal error messages and a RETURN without any computation.

3. Algorithm Usage

3.1 *User Interface.* The input to ADAPT is via the COMMON block INPUTZ except for the function F. This is the most convenient for extensive use of ADAPT but not for occasional use. A subroutine PPFIT4 is provided which has all input as arguments. Entry points PPFIT1, PPFIT2, and PPFIT3 in PPFIT4 have fewer (10, 12, 15, respectively) arguments than PPFIT4 (21 arguments). Due to the variable nature of entry point implementations, these are only indicated by COMMENT cards and local modifications are needed to activate these features.

The basic output is the arrays XKNOTS and COEFS which are arguments to ADAPT (and also the PPFIT subroutines) and the numbers KNOTS and ERROR in the COMMON block RESULZ. The PPFIT routines have KNOTS and ERROR as arguments. In addition the FUNCTION subprogram PPOLY (T, XKNOTS, COEFS, KDIMEN, NDIMEN) returns the value at the point T of the most recently computed approximation. It is automatically available to the user.

There are six levels (−1, 0, 1, 2, 3, 4) of printed output available. LEVEL = −1 only provides fatal error messages; LEVEL = 0 also provides "semifatal" error messages plus 1 line of output; LEVEL = 1 provides a printout of the input and the approximation obtained; LEVEL = 2 provides a condensed trace of the computation and the last two levels are only useful for debugging program modifications.

3.2 *The Characteristic Length of F, CHARF.* The correct operation of ADAPT depends on certain estimates being accurate which, in turn, depend on the relevant subintervals being small enough. Specifically, the sampling that ADAPT does of

F must reveal the nature of F and not allow any significant features to go undetected. The nature of ADAPT is such that for reasonable functions it is very unlikely to miss significant features without using CHARF at all. However, knowing how ADAPT works, one can readily construct examples where it will fail unless CHARF is set properly. The value of CHARF is an *upper* limit on the size of the subintervals for polynomial pieces and it is to be set so that the 4-point Gauss quadrature formulas are reasonably (but not highly) accurate. This means that if F has some complex behavior on a very short segment, then setting CHARF to, say, half the length of this segment will force ADAPT to detect this behavior.

The argument CHARF is essential to proving ADAPT correct (which has not been attempted) but its practical value is debatable. If F is more or less uniformly complicated then short intervals are needed everywhere. It is extremely unlikely, but not impossible, that ADAPT would be fooled in such a case. If F is very smooth except on a very short segment, then ADAPT may well be fooled and setting CHARF small will avoid this. However, it will also force very small intervals where F is smooth and where they are not needed. Thus high reliability is obtained here at the cost of great inefficiency.

The nature of the approximations computed by ADAPT are such that there is a simple and efficient alternative to setting CHARF small. Let [C,D] be the subinterval of [A,B] where F is rough and let F be smooth elsewhere. One can then approximate F on [A,C], [C,D], and [D,B] independently, and the approximations fit together smoothly at C and D to give a single smooth approximation for the entire interval [A,B].

3.3 *The Error Distribution Type, EDIST.* The least squares error for the approximation S(x) to F(x) is

$$E(A,B) = [\int_A^B (F(x) - S(x))^2 \, dx]^{1/2}.$$

To make $E(A,B) < .01$ is equivalent to making $E(A,B)^2 < .0001$, and for any L_p-norm the program actually operates with $ACCUR^p$. Suppose $A=0, B=1$, then we can achieve $E(0,1)^2 < .0001$ by achieving $E(0,1/2)^2 < .0001/2$ and $E(1/2, 1)^2 < .0001/2$ since $E(A,B)^2$ is simply additive over intervals. This approach is called *proportional error distribution*, i.e. the total error requirement is distributed over the subintervals of [A,B] in proportion to the lengths of the subintervals. This choice is selected by EDIST=0 and automatically results in the total error $E(A,B)^2$ less than the specified error $ACCUR^2$.

An alternative approach is to make the errors approximately equal on each of the subintervals independent of their lengths. This is called *fixed error distribution* and is selected by EDIST=2. The argument ACCUR is used for each subinterval and if k subintervals are finally used we see that the total error is then

$$[k * ACCUR^p]^{1/p} = \sqrt[p]{(k)}\ ACCUR$$

and thus ACCUR is not the specified total error when EDIST=2. This alternative is awkward to use because the final approximation error depends on the number of subintervals required which, of course, is unknown until after the approximation is computed. However, for rough or singular F(x) this disadvantage is more than compensated by the superior performance of this error distribution type. This is seen in the theory and verified in actual use.

A compromise approach called *approximate fixed error distribution* is selected by EDIST=1. Basically the algorithm keeps a running estimate of the final number of intervals it will use and adjusts the error requirement for subintervals accordingly. This approach is obviously not foolproof, but the testing reported below shows it to be 98 to 99 percent reliable. In any case, the total error actually obtained is available for the user to see and to test. For smooth, uniformly varying F(x) the proportional error distribution gives perfectly satisfactory efficiency.

We observe that for p = infinity (NORM=3) there is no difference between the fixed and proportional error distributions.

3.4 *Portability Considerations.* Considerable pains have been taken to make the Fortran program portable. It is written in a subset of ANSI Fortran specified by PFORT (see [5]) except that four characters are packed per word rather than one as specified by PFORT. The current version is in single precision and specifically tailored to a machine with a long word length (CDC 6000–7000 series computers). Specific directions are given in the comments for changing the precision or to use it on machines with a shorter word length. In particular, all REAL variables are explicitly declared to facilitate the change to DOUBLE PRECISION. The program has been generated by an experimental Fortran converter which automatically produces versions tailored for different machines and precisions. Several of these versions have been produced and run successfully.

4. Algorithm Testing and Verification

This algorithm is based on a method with theoretically known properties. Care was taken to adhere to the requirements of that theory and considerable analysis has been made of various features and parts of the algorithm. However, no formal proof has been attempted since, as with many numerical algorithms, one cannot say a priori what is to be computed. A somewhat related algorithm has been proved correct with a few idealizing assumptions (infinite precision arithmetic and infinite memory, for example); see [3, 4].

Very extensive testing of the algorithm has been performed to see if the theoretical expectations are, in fact, realized by this algorithm. There are some approximation methods where this has not been the case. These tests are discussed at some length in [2] and we summarize them by saying that this algorithm performs as expected from the theory. The results in [2] give many insights into the practical use of this algorithm. About 2500 to 3000 different approximations have been computed in these tests and all have been examined for signs of incorrect performance. A few hundred of these runs were specifically designed to test the validity and correctness of the program.

The code with the ADAPT includes a set of 20 test functions (10 of them parame-

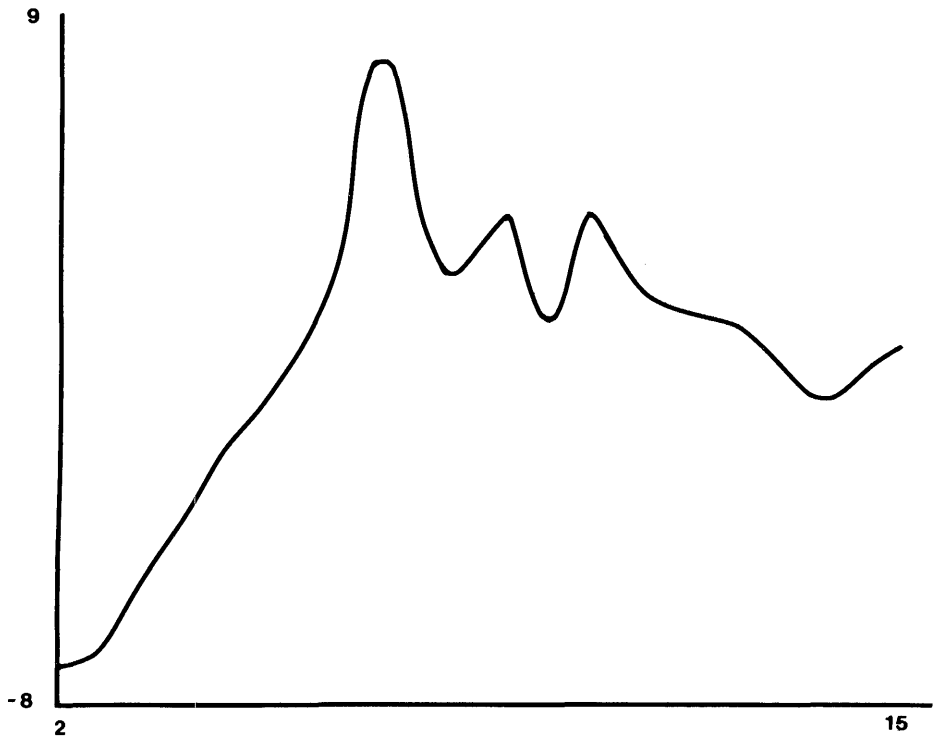


Fig. 1. The complicated function with seven pieces approximated in the second example

terized in various ways) and a driver to exercise ADAPT and measure its performance in various ways. These are included because it is felt they will be useful for testing other function approximation programs. We note that the 20 test functions are chosen to present various features of the approximation problem and we found it amazingly difficult to get the first few derivatives of these functions computed correctly.

We give sample output for two functions, one easy and one difficult. The first is $F(x) = 1./(1+(x-2.5)^4)$ with LEVEL=2 output given. The second function is shown in Figure 1 with LEVEL = 0 output given. In the first example the first and last three lines of output are from the test driver and the specific information about the problem is printed by ADAPT. For the second example the approximation is on [2,15] for polynomial degree 6 with 1 continuous derivative; the requested and estimated accuracies are .001 and .00057 in the minimax norm; proportional error distribution was used with CHARF=4. This example is function 18 of the set of test functions and one of its parameters, P8F, has been set to 1.6 here instead of 0.6 as given in the DATA for F(x). The output has been reformatted to accommodate the narrowed page width here.

```
*****ADAPT FOR FUNCTION 4 = HUMP AT 2.5 RECIPROCAL OF QUARTIC
PIECEWISE POLYNOMIAL APPROXIMATION ON INTERVAL 1.0000E+00 5.0000E+00
OF DEGREE 5 WITH 2 CONTINUOUS DERIVATIVES
ACCURACY REQUESTED IS 1.0000E-03 MEASURED BY LEAST SQUARES
OTHER INPUT/DEFAULT VARIABLES ARE FOSCIL = 4.0000E+00
EMEAS = 2.0000E+00 ---- PROPORTIONAL ERROR DISTRIBUTION
KNOT 2 AT 1.50000E+00, LOCAL-GLOBAL ERRORS = 2.6008E-09  5.0998E-05
KNOT 3 AT 2.00000E+00, LOCAL-GLOBAL ERRORS = 3.5626E-08  1.9552E-04
KNOT 4 AT 2.50000E+00, LOCAL-GLOBAL ERRORS = 2.5115E-08  2.5168E-04
KNOT 5 AT 3.00000E+00, LOCAL-GLOBAL ERRORS = 2.5115E-08  2.9742E-04
KNOT 6 AT 3.50000E+00, LOCAL-GLOBAL ERRORS = 3.5626E-08  3.5225E-04
KNOT 7 AT 4.00000E+00, LOCAL-GLOBAL ERRORS = 2.6008E-09  3.5593E-04
KNOT 8 AT 5.00000E+00, LOCAL-GLOBAL ERRORS = 2.4834E-09  3.5940E-04

--- ADAPTIVE PIECEWISE POLYNOMIAL APPROXIMATION OF DEGREE 5 WITH
2 CONTINUOUS DERIVATIVES NEEDED 8 KNOTS FOR ERROR = 3.5940E-04
KNOT LOCATION      X-POWER COEFFICIENTS RELATIVE TO KNOT LOCA-
                     TIONS
1  1.000000000000E+00   1.649484536082E-01
                        3.673078966947E-01
                        4.506148423367E-01
                        2.850451366482E-01
                        5.027123624777E-01
                       -9.058008728205E-01
2  1.500000000000E+00    5.000000000000E-01
                         1.000000000000E+00
                         5.000000000000E-01
                        -1.032973743131E+00
                        -2.117239975575E+00
                         2.484021982495E+00
3  2.000000000000E+00    9.411764705882E-01
                         4.429065743945E-01
                        -1.120293099939E+00
                         7.978831671079E-01
                         7.848565031541E-01
                        -1.003053124363E+00
4  2.500000000000E+00    1.000000000000E+00
                                         0
                                         0
                         1.400366374921E-01
                        -1.722776307754E+00
                         1.003053124363E+00
5  3.000000000000E+00    9.411764705882E-01
                        -4.429065743945E-01
                        -1.120293099939E+00
                        -9.426012619580E-01
                         4.092814980663E+00
```

```
                        −2.484021982495E+00
6  3.500000000000E+00    5.000000000000E−01
                        −1.000000000000E+00
                         5.000000000000E−01
                         9.740323204476E−01
                        −1.761789819574E+00
                         9.058008728205E−01
7  4.000000000000E+00    1.649484536082E−01
                        −3.673078966947E−01
                         4.506148423367E−01
                        −3.547233211608E−01
                         1.658371414434E−01
                        −3.440822109296E−02

ADAPT USED 117 FUNCTION VALUES FOR ERRORS
SPECIFIED = 1.00000E−03 ESTIMATED BY ADAPT = 3.59399E−04
INDEPENDENT CHECK = 3.05041E−04 TIME USED = .17200 SECONDS

*****ADAPT FOR FUNCTION 18 = COMPLICATED FUNCTION WITH 7 PIECES
-------ADAPTIVE PIECEWISE POLYNOMIAL APPROXIMATION OF DEGREE 6 WITH
1 CONTINUOUS DERIVATIVES NEEDED 30 KNOTS FOR ERROR = 5.7301E−04
```

5. High Level Expression of the Algorithm ADAPT

The following description of ADAPT indicates its basic structure and methods of computation:

```
PROCEDURE PPFIT − WITH ARGUMENTS =
  F         .FUNCTION TO FIT          LEVEL    .OUTPUT LEVEL −1 TO 2
  A,B       .INTERVAL ENDPTS          CHARF    .LIMIT ON PIECES LENGTH
  ACCUR     .ACCURACY DESIRED         EDIST    .TYPE OF ERROR CONTROL
  DEGREE    .POLYNOMIAL DEGREE        NBREAK   .NUMBER OF SPECIFIED
  SMOOTH    .NO. CONT. DERIVS                   BREAK POINTS IN FIT.
  NORM      .MEAS. OF L−P ERROR                 HAS RELATED ARGUMENTS
              UP IN (0, INFINITY)               GIVING DETAILS.

  OUTPUT = KNOTS.                               .NUMBER OF KNOTS
           ERROR.                               .ERROR ACHIEVED
           XKNOTS(K),K=1 TO KNOTS               .KNOT LOCATIONS
           COEFS(K,N),K=1 TO KNOTS              .POWER COEF. RELATIVE
                      N=1 TO DEGREE+1           .TO THE KNOT LOCATIONS

*** THIS PROGRAM MERELY COMPUTES A FEW DEFAULT VALUES
    AND PUTS VARIABLES IN COMMON BLOCKS, ETC

    CALL ADAPT − TO DO THE APPROXIMATION
END PPFIT
SUBPROGRAM ADAPT
  CALL SETUP − CHECK INPUT, INITIALIZE THINGS, PRINT PROBLEM

  *** LOOP OVER PROCESSING INTERVALS ***

  CALL TAKE    − AN INTERVAL OFF THE STACK
  CALL COMPUT  − AN APPROX ON THIS INTERVAL
  CALL CHECK   − FOR DISCARDING OR DIVIDING INTERVAL
  CALL PUT     − NEW INTERVALS ON STACK, UPDATE ALGORITHM STATUS
  CALL TERMIN  − TEST FOR FINISH, PRINT INTERMEDIATE OUTPUT
  IF NOT FINISHED − REPEAT LOOP

  CALL SUMMARY − FOR FINAL OUTPUT
END ADAPT

SUBPROGRAM SETUP
  SET LIMITS ON COMPUTATION PARAMETERS
  CHECK ALL INPUT DATA
  INITIALIZE VARIABLES AND INTERVAL STACK
  PRINT PROBLEM STATEMENT
END SETUP

SUBPROGRAM TAKE
  CHECK FOR BREAK POINT IN TOP INTERVAL
```

```
    IF SO – ADJUST XKNOTS TO MAKE IT A PARTITION POINT
    ELSE – DO NOTHING
END TAKE

SUBPROGRAM PUT
  CHECK FOR DISCARDING INTERVAL
    IF SO – UPDATE ERROR ESTIMATE
            ADJUST STACK
            CALL PTRANS – TO OBTAIN COEFS FOR THIS INTERVAL
            UPDATE XKNOTS AND COEFS
    ELSE – SUBDIVIDE INTERVAL AND PLACE 2 NEW ONES ON STACK
            CHECK FOR EXCEEDING MAX STACK SIZE OR OBTAINING
              AN INTERVAL WHICH IS TOO SHORT. SUCH SHORT INTERVALS
              ARE DISCARDED WITHOUT REGARD TO ERROR CONTROL POLICY
              AND WITH MESSAGE
END PUT

  SUBPROGRAM PTRANS – OF PUT
    CHANGES POLYNOMIAL REPRESENTATION FROM NEWTON DIVIDED
    DIFFERENCE FORM TO POWER FORM WITH ORIGIN SHIFTED TO THE
    XKNOT VALUE ON LEFT OF INTERVAL. USES SYNTHETIC DIVISION
  END PTRANS

SUBPROGRAM COMPUT
  OBTAIN       – VALUES OF F AND DERIVATIVES. MAKE ADJUSTMENTS
                 IF A BREAK POINT IS INVOLVED
  CALL NEWTON – FOR DIVIDED DIFFERENCES OF INTERPOLATING
                 POLYNOMIAL FOR THIS INTERVAL
  CALL ERRINT – TO ESTIMATE LOCAL ERROR + (F(X)–POLYDD(X))**P
                 FOR L–P NORM, 0 LT P LE INFINITY
END COMPUT

  SUBPROGRAM NEWTON – OF COMPUT
     BUILD UP TRUE DIVIDED DIFFERENCE TABLE WITH MULTIPLE POINTS
     AT THE INTERVAL ENDS PLUS INTERPOLATION POINTS
  END NEWTON

  SUBFUNCTION POLYDD – OF COMPUT
     EVALUATES POLYNOMIAL FROM DIFFERENCE TABLE OUT OF NEWTON
                FOR L–P NORM WITH P IN (0, INFINITY)
  END POLYDD

  SUBPROGRAM ERRINT – OF COMPUT
     USES 4-POINT GAUSS QUADRATURE TO ESTIMATE ERROR NORM ON
     INTERVAL. SPECIAL COMPUTATION FOR MAX–NORM, P=INFINITY.
  END ERRINT

SUBPROGRAM CHECK
  USES ERROR DISTRIBUTION TYPE AND CHARF TO DECIDE ON DISCARD
END CHECK

SUBPROGRAM TERMIN
  PRINT – INTERMEDIATE OUTPUT, IF ANY REQUESTED BY LEVEL = 2
  TEST  – FOR TERMINATION EMPTY STACK – NORMAL
                              EXCEEDED XKNOTS LIMIT – ABNORMAL
END TERMIN

SUBPROGRAM SUMMARY
  LEVEL = –1  NOTHING
        =  0  1 LINE (OUTPUT RETURNED IN COMMON, ARGUMENTS)
        =  1  KNOTS AND COEFFICIENTS
        =  2  DITTO
END SUMMARY

FUNCTION PPOLY
  THE PIECEWISE POLYNOMIAL COMPUTED BY PREVIOUS CALL ON PPFIT
END PPOLY
```

REFERENCES

1. RICE, J.R. Adaptive approximation. *J. Approx. Theory 16* (1976), 329-337.
2. RICE, J.R. On adaptive piecewise polynomial approximation. In *Theory of Approximation*, A.G. Law and B.N. Sahney, Ed., Academic Press, New York, 1976, pp. 359-386.
3. RICE, J.R. Parallel algorithms for adaptive quadrature II—Metalgorithm correctness. *Acta Inform. 4* (1975), 273-285.
4. RICE, J.R. Parallel algorithms for adaptive quadrature III—Program correctness. *ACM Trans. Math. Software 2*, 1 (March 1976), 1-30.
5. RYDER, B.G. The PFORT verifier. *Software—Practice and Experience 4* (1974), 359-377.

ALGORITHM

[The listing printed here is an abridged version of the complete algorithm, which is available upon request from the ACM Algorithms Distribution Service.]

```
      SUBROUTINE ADAPT(F, XKNOTS, COEFS, KDIMEN, NDIMEN)                ADA   1Ø
C                                                                       ADA   2Ø
C     THIS ALGORITHM COMPUTES A PIECEWISE POLYNOMIAL APPROXIMATION      ADA   3Ø
C  OF SPECIFIED SMOOTHNESS, ACCURACY AND DEGREE.  THE INPUT TO THE      ADA   4Ø
C  COMPUTATION IS                                                       ADA   5Ø
C                                                                       ADA   6Ø
C   F       - FUNCTION BEING APPROXIMATED. IT MUST PROVIDE VALUES OF    ADA   7Ø
C             DERIVATIVES UP TO THE ORDER OF SMOOTHNESS SPECIFIED FOR   ADA   8Ø
C             THE APPROXIMATION.  THE CALLING SEQUENCE IS F(X,FDERV) AND ADA   9Ø
C             FDERV CONTAINS THE DERIVATIVES( SEE CONSTRAINT BELOW)     ADA  1ØØ
C   A,B     - THE ENDPOINTS OF THE INTERVAL OF APPROXIMATION            ADA  11Ø
C   ACCUR   - THE ACCURACY REQUIRED FOR THE APPROXIMATION               ADA  12Ø
C   SMOOTH  - THE SMOOTHNESS REQUIRED FOR THE APPROXIMATION             ADA  13Ø
C               = Ø  MEANS CONTINUOUS                                   ADA  14Ø
C               = 1  MEANS CONTINUOUS SLOPE                             ADA  15Ø
C               = 2  MEANS CONTINUOUS SECOND DERIVATIVE, ETC.           ADA  16Ø
C   DEGREE  - THE DEGREE OF THE POLYNOMIAL PIECES.                      ADA  17Ø
C             MUST HAVE DEGREE GT 2*SMOOTH                              ADA  18Ø
C                                                                       ADA  19Ø
C     *****  ***** SECONDARY INPUT - ITEMS WITH DEFAULT VALUES POSSIBLE ADA  2ØØ
C   CHARF   - CHARACTERISTIC LENGTH OF THE FUNCTION F(X). PIECES ARE NOT ADA  21Ø
C             LONGER THAN THIS LENGTH.                                  ADA  22Ø
C                   DEFAULT=(B-A) IF DEGREE GT 1, ELSE (B-A)/3          ADA  23Ø
C   NORM    - NORM TO MEASURE THE APPROXIMATION ERROR                   ADA  24Ø
C               = 1  L1 APPROXIMATION (LEAST DEVIATIONS)                ADA  25Ø
C               = 2  L2 APPROXIMATION (LEAST SQUARES)                   ADA  26Ø
C               = 3  TCHEBYCHEFF (MINIMAX) APPROXIMATION                ADA  27Ø
C               =-P  (NEGATIVE VALUE) GENERAL LP APPROXIMATION          ADA  28Ø
C                   DEFAULT= 2                                          ADA  29Ø
C   NBREAK  - NUMBER OF SPECIAL BREAK POINTS IN THE APPROXIMATION.      ADA  3ØØ
C             ASSOCIATED INPUT VARIABLES ARE                            ADA  31Ø
C               XBREAK(J)  - LOCATION OF BREAK POINTS                   ADA  32Ø
C               DBREAK(J)  - DERIVATIVE BROKEN AT XBREAK                ADA  33Ø
C               BLEFT (J)  - VALUE FROM LEFT FOR DBREAK DERIVATIVE      ADA  34Ø
C               BRIGHT(J)  -  -   -  RIGHT -   -       -               ADA  35Ø
C                   DEFAULT = Ø                                         ADA  36Ø
C   LEVEL   - LEVEL OF OUTPUT FROM ADAPT                                ADA  37Ø
C               = -1 NO OUTPUT AT ALL EXCEPT FOR FATAL INPUT ERRORS     ADA  38Ø
C               =  Ø  ERROR CONDITIONS NOTED, FINAL SUMMARY             ADA  39Ø
C               =  1  PRINT THE APPROXIMATION OUT                       ADA  4ØØ
C               =  2  DETAILS OF THE COMPUTATION                        ADA  41Ø
C               =  3 DEBUG OUTPUT,  = 4 LOTS OF DEBUG OUTPUT            ADA  42Ø
C                    DEFAULT = Ø                                        ADA  43Ø
C   EDIST   - SWITCH TO CHANGE FROM PROPORTIONAL ERROR DISTRIBUTION     ADA  44Ø
C             TO FIXED DISTRIBUTION. THIS IS PRIMARILY OF USE IN        ADA  45Ø
C             APPROXIMATION OF FUNCTIONS WITH SINGULARITIES. ONE SHOULD ADA  46Ø
C             USE NORM = 1. OR SO IN SUCH CASES                         ADA  47Ø
C               = Ø  PROPORTIONAL DISTRIBUTION                          ADA  48Ø
C               = 1  APPROXIMATE FIXED ERROR DISTRIBUTION               ADA  49Ø
C                    ATTEMPTS TO ACHIEVE SPECIFIED ACCURACY VALUE ACCUR  ADA  5ØØ
C               = 2  TRUE FIXED ERROR DISTRIBUTION                      ADA  51Ø
C                                                                       ADA  52Ø
C     THE OUTPUT OF THE COMPUTATION CONSISTS OF 3 PARTS, EACH RETURNED  ADA  53Ø
C     TO THE USER IN A DIFFERENT WAY. THEY ARE                          ADA  54Ø
C                                                                       ADA  55Ø
C   XKNOTS,COEFS - ARRAYS DEFINING THE PIECEWISE POLYNOMIAL RESULT.     ADA  56Ø
C            XKNOTS(K)  = KNOTS OF THE APPROXIMATION ( K = 1 TO KNOTS)  ADA  57Ø
C                         THE LAST ONE IS RIGHT END POINT OF INTERVAL   ADA  58Ø
C            COEFS(K,N) = COEFFICIENT OF (X - XKNOT(K))**(N-1) IN THE   ADA  59Ø
```

```
C                        INTERVAL XKNOT(K) TO XKNOT(K+1)                ADA  6ØØ
C                          K = 1 TO KNOTS-1  AND  N = 1 TO DEGREE+1     ADA  61Ø
C           THESE ARRAYS ARE PASSED AS ARGUMENTS SO AS TO USE VARIABLE   ADA  62Ø
C           DIMENSIONS.                                                  ADA  63Ø
C              KDIMEN - DIMENSION USED FOR  XKNOTS IN CALLING PROGRAM    ADA  64Ø
C              NDIMEN - COEFS IS DECLARED  COEFS(KDIMEN,NDIMEN) IN THE   ADA  65Ø
C                       CALLING PROGRAM.                                ADA  66Ø
C              ***** NOTE ***** SEVERAL SMALL ARRAYS HERE HAVE FIXED      ADA  67Ø
C                       DIMENSIONS THAT LIMIT DEGREE AND THUS NDIMEN     ADA  68Ø
C                       SHOULD NOT EXCEED THIS LIMIT (CURRENTLY = 12)    ADA  69Ø
C                                                                        ADA  7ØØ
C   PPOLY  - THE PIECEWISE POLYNOMIAL APPROXIMATING FUNCTION.            ADA  71Ø
C           THIS FUNCTION SUBPROGRAM IS AVAILABLE TO THE USER AT THE     ADA  72Ø
C           COMPLETION OF ADAPT.                                         ADA  73Ø
C                                                                        ADA  74Ø
C   RESULZ - A LABELED COMMON BLOCK WITH  ERROR,KNOTS  IN IT             ADA  75Ø
C             KNOTS - NUMBER OF KNOTS OF PPOLY                           ADA  76Ø
C             ERROR - ACCURACY OF PPOLY AS ESTIMATED BY ADAPT            ADA  77Ø
C                                                                        ADA  78Ø
C  ********** DIMENSION CONSTRAINTS **********                          ADA  79Ø
C     MAXKNT - MAX NUMBER OF KNOTS TAKEN FROM USER VIA KDIMEN            ADA  8ØØ
C              ARRAYS WITH THIS DIMENSION                                ADA  81Ø
C                   COEFS   XKNOTS                                       ADA  82Ø
C     MAXPAR - MAX NUMBER OF PARAMETERS PER INTERVAL ( = 12 CURRENTLY )  ADA  83Ø
C                   USER PROVIDED NDIMEN MUST HAVE NDIMEN LE MAXPAR      ADA  84Ø
C                   MUST HAVE  DEGREE + 1 LE MAXPAR                      ADA  85Ø
C              ARRAYS WITH THIS DIMENSION (OR RELATED VALUES )           ADA  86Ø
C                D       DDTEMP  FDERVL  FDERVR  FDUMB   FACTOR         ADA  87Ø
C                FINTRP  FLEFT   FRIGHT  POWERS  XTEMP   XINTRP  XDD    ADA  88Ø
C              ***** NOTE ***** MAXPAR ALSO AFFECTS ARGUMENT FDERV       ADA  89Ø
C              OF FUNCTION F.  FDERVL, FDERVR ARE ALSO INVOLVED.         ADA  9ØØ
C              SHOULD DECLARE FDERV OF SIZE 6 IN F TO BE SAFE.           ADA  91Ø
C     MAXAUX - MAXIMUM NUMBBER OF AUXILIARY INPUT( = 2Ø NOW ). ARRAYS    ADA  92Ø
C                XBREAK  DBREAK  BLEFT   BRIGHT                          ADA  93Ø
C     MAXSTK - MAX SIZE OF ACTIVE INTERVAL STACK                         ADA  94Ø
C              MIN INTERVAL LENGTH IS 2**(-MAXSTK)*(B-A). ARRAYS         ADA  95Ø
C                 XLEFT   XRIGHT                                         ADA  96Ø
C                                                                        ADA  97Ø
C  **********  PORTABILITY CONSIDERATIONS  **********                    ADA  98Ø
C                                                                        ADA  99Ø
C     THIS PROGRAM IS IN ANSI STANDARD FORTRAN.  IN ADDITION, IT MEETS   ADA 1ØØØ
C     ALL THE REQUIREMENTS OF THE BELL LABS PORTABLE FORTRAN -PFORT-     ADA 1Ø1Ø
C     EXCEPT ONE.  HOLLERITH CHARACTERS ARE PACKED 4/WORD RATHER THAN    ADA 1Ø2Ø
C     1/WORD AS SPECIFIED BY PFORT.                                      ADA 1Ø3Ø
C     NEVERTHELESS, THIS PROGRAM IS AFFECTED IN SEVERAL WAYS BY A        ADA 1Ø4Ø
C     CHANGE IN MACHINE WORD LENGTH AND CHANGING TO DOUBLE PRECISION.    ADA 1Ø5Ø
C     ***** THIS VERSION IS FOR THE MACHINE WITH THE LONGEST SINGLE      ADA 1Ø6Ø
C           PRECISION WORD (CDC).  THE LENGTH OF SOME CONSTANTS IN       ADA 1Ø7Ø
C           THE SUBPROGRAM COMPUT EXCEEDS THE CAPACITY OF SOME FORTRAN   ADA 1Ø8Ø
C           COMPILERS AND CAN PREVENT COMPILATION.                       ADA 1Ø9Ø
C  INPUT-OUTPUT -- IS OF THREE TYPES.                                    ADA 11ØØ
C        FATAL ERROR MESSAGES - OCCUR IN SETUP,TAKE,PUT AND TERMIN       ADA 111Ø
C                               THEY CANNOT BE SWITCHED OFF              ADA 112Ø
C        USER AND DEBUGGING OUTPUT - CAN BE SWITCHED OFF                 ADA 113Ø
C     THESE INVOLVE MANY FORMATS LIKE  E15.8, F12.8, E22.13, ETC.        ADA 114Ø
C     SOME FORTRAN COMPILERS REQUIRE D-FORMAT FOR DOUBLE PRECISION.      ADA 115Ø
C     SOME DO NOT HANDLE E22.13 ON MACHINES OF SHORTER WORD LENGTH.      ADA 116Ø
C                                                                        ADA 117Ø
C     SUMARY PRINTS COEFFICIENTS 5 PER LINE, 6 PER LINE IS BETTER        ADA 118Ø
C     FOR SHORTER WORD LENGTHS.  DOUBLE PRECISION ON MANY MACHINES       ADA 119Ø
C     LIMITS ONE TO 4 PER LINE.                                          ADA 12ØØ
C                                                                        ADA 121Ø
C  CONSTANTS -- THE GAUSS WEIGHTS AND ABSCISSA IN COMPUTE ARE GIVEN      ADA 122Ø
C               TO 15 DIGITS IN THE COMMENTS                             ADA 123Ø
C                                                                        ADA 124Ø
C  DOUBLE PRECISION CONVERSION -- REQUIRES FOUR STEPS                    ADA 125Ø
C     1. ALL REAL VARIABLES ARE DECLARED DOUBLE PRECISION. THIS IS       ADA 126Ø
C        DONE BY CHANGING REAL TO DOUBLE PRECISION AS ALL REALS ARE      ADA 127Ø
C        EXPLICITLY DECLARED AND ROOM IS LEFT FOR THIS CHANGE.  REAL     ADA 128Ø
C        VARIABLES APPEAR BEFORE INTEGERS IN ALL COMMON BLOCKS.          ADA 129Ø
C                                                                        ADA 13ØØ
C     2. ADD D TO CONSTANTS( E.G. 1.Ø = 1.ØDØ ). ADJUST LENGTH OF       ADA 131Ø
C        GAUSS WEIGHTS IN COMPUTE.                                       ADA 132Ø
C                                                                        ADA 133Ø
C     3. CHANGE ABS,AMAX1,FLOAT  AT MANY PLACES                          ADA 134Ø
C                                                                        ADA 135Ø
```

```
C     4. ADJUST THE INTERVAL STACK SIZE = DIMENSIONS OF XLEFT, XRIGHT   ADA 136Ø
C        AND VALUE OF MAXSTK.  ADJUST THE VALUE BUFFER IN PUT TO BE     ADA 137Ø
C        ABOUT .1E-K FOR A MACHINE WITH K+2 DECIMAL DIGITS.             ADA 138Ø
C                                                                       ADA 139Ø
      REAL A, ACCUR, B, BLEFT, BRIGHT, CHARF, DDTEMP, DSCTOL, ERROR,    ADA 14ØØ
     * ERRORI, FACTOR, FINTRP, FLEFT, FRIGHT, NORM, XBREAK, XDD,        ADA 141Ø
     * XINTRP, XLEFT, XRIGHT                                           ADA 142Ø
      DIMENSION XBREAK(2Ø), DBREAK(2Ø), BLEFT(2Ø), BRIGHT(2Ø)           ADA 143Ø
      DIMENSION XLEFT(5Ø), XRIGHT(5Ø), FACTOR(12), FMESGE(6)            ADA 144Ø
      DIMENSION DDTEMP(12,12), FINTRP(1Ø), FLEFT(6), FRIGHT(6),         ADA 145Ø
     * XDD(12), XINTRP(1Ø)                                             ADA 146Ø
      INTEGER BOTH, BREAK, DBREAK, DEGREE, EDIST, FMESGE, RIGHT,        ADA 147Ø
     * RIGHTX, SMOOTH                                                  ADA 148Ø
      LOGICAL DISCRD, FATAL, FINISH                                     ADA 149Ø
      REAL XKNOTS(KDIMEN), COEFS(KDIMEN,NDIMEN)                         ADA 15ØØ
      EXTERNAL F                                                        ADA 151Ø
      REAL F                                                            ADA 152Ø
C                                                                       ADA 153Ø
      COMMON /INPUTZ/ A, B, ACCUR, NORM, CHARF, XBREAK, BLEFT, BRIGHT,  ADA 154Ø
     * DBREAK, DEGREE, SMOOTH, LEVEL, EDIST, NBREAK, KNTDIM, NPARDM     ADA 155Ø
C                     KNTDIM - KDIMEN, NAME CHANGED TO PUT IN COMMON    ADA 156Ø
C                     NPARDM - NDIMEN, NAME CHANGED TO PUT IN COMMON    ADA 157Ø
      COMMON /RESULZ/ ERROR, KNOTS                                      ADA 158Ø
C                     KNOTS = FINAL NO. OF KNOTS, INCLUDES B AS ONE.    ADA 159Ø
C                     ERROR = ESTIMATE OF ERROR ACTUALLY ACHIEVED.      ADA 16ØØ
      COMMON /KONTRL/ DSCTOL, ERRORI, XLEFT, XRIGHT, BREAK, BOTH,       ADA 161Ø
     * FACTOR, FMESGE, IBREAK, INTERP, LEFT, MAXAUX, MAXKNT, MAXPAR,    ADA 162Ø
     * MAXSTK, NPAR, NSTACK, RIGHT, DISCRD, FATAL, FINISH               ADA 163Ø
C             KONTRL CONTAINS GENERALLY USEFUL VARIABLES                ADA 164Ø
C                     FATAL  - SWITCH FOR DETECTION OF FATAL ERROR      ADA 165Ø
C                              INCLUDING EXCESSIVE INTERVAL SUBDIVISIONADA 166Ø
C                              WHICH DOES NOT TERMINATE THE COMPUTATIONADA 167Ø
C                     FINISH - SWITCH TO TERMINATE ALGORITHM            ADA 168Ø
C                     MAXS   - SEE COMMENTS EARLIER                     ADA 169Ø
C                     NSTACK - COUNTER FOR INTERVAL STACK, CONSISTS OF  ADA 17ØØ
C                              (XLEFT(J),XRIGHT(J))  J = 1 TO NSTACK    ADA 171Ø
C                     ERRORI - ERROR ESTIMATE FOR TOP INTERVAL          ADA 172Ø
C                     DSCTOL - TOLERANCE TO CHECK DISCARDING INTERVALS  ADA 173Ø
C                     DISCRD - SWITCH TO SIGNAL DISCARD OF TOP INTERVAL ADA 174Ø
C                     FACTOR - ARRAY OF FACTORIALS                      ADA 175Ø
C                     NPAR   - NUMBER OF PAREMETERS = DEGREE + 1        ADA 176Ø
C                     FMESGE - STRING =  **** FATAL ERROR *****         ADA 177Ø
C                     INTERP - NUMBER OF INTERIOR INTERPOLATION POINTS  ADA 178Ø
C                              IN THE NORMAL INTERVAL                   ADA 179Ø
C                     IBREAK - COUNTER ON BREAK POINTS                  ADA 18ØØ
C                     BREAK  - SWITCH FOR BREAK POINT IN TOP INTERVAL   ADA 181Ø
C                              Ø     = NO BREAK PRESENT                 ADA 182Ø
C                              LEFT  = BREAK AT XLEFT(NSTACK)           ADA 183Ø
C                              RIGHT = BREAK AT XRIGHT(NSTACK)          ADA 184Ø
C                              BOTH  = BREAK AT BOTH ENDS               ADA 185Ø
      COMMON /COMDIF/ DDTEMP, FINTRP, FLEFT, FRIGHT, XDD, XINTRP,       ADA 186Ø
     * LEFTX, NINTRP, RIGHTX                                           ADA 187Ø
C             COMDIF CONTAINS VARIABLES USED ONLY BY COMPUT AND FRIENDS.ADA 188Ø
C                     NINTRP - NUMBER OF INTERIOR INTERPOLATION POINTS  ADA 189Ø
C                              FOR THE CURRENT INTERVAL                 ADA 19ØØ
C                     XINTRP - INTERIOR INTERPOLATION POINTS            ADA 191Ø
C                     FINTRP - F VALUES AT XINTRP POINTS                ADA 192Ø
C                     LEFTX  - MULTIPLICITY OF INTERPOLATION AT XLEFT   ADA 193Ø
C                              = NO. OF DERIVATIVES MATCHED AT XLEFT    ADA 194Ø
C                     FLEFT  - VALUES OF F AND ITS DERIVATIVES AT XLEFT ADA 195Ø
C                     RIGHTX - MULTIPLICITY OF INTERPOLATION AT XRIGHT  ADA 196Ø
C                     FRIGHT - VALUES OF F AND DERIVATIVES AT XRIGHT    ADA 197Ø
C                     DDTEMP - THE ARRAY OF DIVIDED DIFFERENCES         ADA 198Ø
C                     XDD    - THE X VALUES FOR DDTEMP WITH PROPER      ADA 199Ø
C                              MULTIPLICITIES OF XLEFT AND XRIGHT       ADA 2ØØØ
      COMMON /SAVEIT/ IKNOT                                             ADA 2Ø1Ø
C                                                                       ADA 2Ø2Ø
C---------------------- MAIN CONTROL PROGRAM ------------------------   ADA 2Ø3Ø
C                                                                       ADA 2Ø4Ø
C   ***** NOTE - ARGUMENTS BELOW ARE FOR READABILITY ONLY    *****      ADA 2Ø5Ø
C   *****        EXCEPT FOR F AND XKNOTS,COEFS,KDIMEN,NDIMEN *****     ADA 2Ø6Ø
C                                                                       ADA 2Ø7Ø
C                  CHECK INPUT, INITIAL COMPUTATIONS, PRINT PROBLEM     ADA 2Ø8Ø
C                                                                       ADA 2Ø9Ø
      CALL SETUP(XKNOTS, COEFS, KDIMEN, NDIMEN)                         ADA 21ØØ
C                                                                       ADA 211Ø
C         CHECK FOR FATAL ERROR IN PROBLEM SPECIFICATION                ADA 212Ø
      IF (FATAL) RETURN                                                 ADA 213Ø
```

```
C                                                                       ADA 214Ø
C                    LOOP OVER PROCESSING OF INTERVALS                  ADA 215Ø
   1Ø CALL TAKE(INTERV)                                                 ADA 216Ø
C                                                                       ADA 217Ø
      CALL COMPUT(F, APPROX, INTERV)                                    ADA 218Ø
C                                                                       ADA 219Ø
C                    CHECK FOR DISCARDING INTERVALS                     ADA 22ØØ
      CALL CHECK(FUNCT, CHARCT)                                         ADA 221Ø
C                                                                       ADA 222Ø
C           PUT NEW INTERVALS ON STACK OR DISCARD, UPDATE STATUS        ADA 223Ø
      CALL PUT(INTERV, XKNOTS, COEFS, KDIMEN, NDIMEN)                   ADA 224Ø
C                                                                       ADA 225Ø
      CALL TERMIN(TEST, AND, PRINT, XKNOTS, KDIMEN)                     ADA 226Ø
C                                                                       ADA 227Ø
      IF (.NOT.FINISH) GO TO 1Ø                                         ADA 228Ø
C                                                                       ADA 229Ø
      CALL SUMARY(XKNOTS, COEFS, KDIMEN, NDIMEN)                        ADA 23ØØ
      RETURN                                                            ADA 231Ø
      END                                                               ADA 232Ø


      SUBROUTINE PPFIT4(F, XLFT, XRGT, EPSLN, NP1ECE, ERREST, XKNOTS,   PPF   1Ø
     * COEFS, KDIMEN, NDIMEN, NDEG, NSMTH, EMEAS, LPRNT, FOSCIL, ATYPE, PPF   2Ø
     * KBREAK, BRAKPT, KDERVB, VALLFT, VALRGT)                          PPF   3Ø
C                                                                       PPF   4Ø
C       ================================================================PPF   5Ø
C                                                                       PPF   6Ø
C **  THIS SET OF FOUR CONTROL PROGRAMS SET VARYING NUMBERS OF DEFAULT   PPF   7Ø
C     VALUES FOR ARGUMENTS.  IT USES ENTRY STATEMENTS WHICH ARE DONE    PPF   8Ø
C     DIFFERENTLY BY VARIOUS FORTRANS.  FOR THIS REASON ENTRY STATEMENTSPPF   9Ø
C     ARE ONLY INDICATED BY COMMENT CARDS.  WRITING FOUR SEPARATE ROU-  PPF  1ØØ
C     TINES APPROXIMATELY TRIPLES THE LENGTH OF THE TOTAL CODE.         PPF  11Ø
C                                                                       PPF  12Ø
C     THE FOLLOWING TABULATES THE INTERNAL AND EXTERNAL NAMES OF THE    PPF  13Ø
C     ARGUMENTS ALONG WITH THEIR DEFAULT VALUES FOR THE VARIOUS PPFIT.  PPF  14Ø
C     NOTE THAT THIS ALLOWS THE ARGUMENTS TO BE PUT INTO A COMMON BLOCK PPF  15Ø
C     AND AVOIDS LONG ARGUMENT LISTS WITHIN ADAPT ITSELF.               PPF  16Ø
C                                                                       PPF  17Ø
C                                                                       PPF  18Ø
C     INTERNAL   EXTERNAL     DEFAULT VALUE    SET BY PPFIT NUMBER      PPF  19Ø
C      F          F              NONE                                  PPF  2ØØ
C      A,B        A,B            NONE                                  PPF  21Ø
C      ACCUR      EPSLN          NONE                                  PPF  22Ø
C      KNOTS      NPIECE         OUTPUT                                PPF  23Ø
C      ERROR      ERREST         OUTPUT                                PPF  24Ø
C      XKNOTS     XKNOTS         OUTPUT                                PPF  25Ø
C      COEFS      COEFS          OUTPUT                                PPF  26Ø
C      KDIMEN     KDIMEN         NONE                                  PPF  27Ø
C      NDIMEN     NDIMEN         NONE                                  PPF  28Ø
C      DEGREE     NDEG           3                1                    PPF  29Ø
C      SMOOTH     NSMTH          Ø                1                    PPF  3ØØ
C      NORM       EMEAS          3                1 2                  PPF  31Ø
C      LEVEL      LPRNT          1                1 2                  PPF  32Ø
C      CHARF      FOSCIL         VARIABLE         1 2                  PPF  33Ø
C      EDIST      ATYPE          1                1 2 3                PPF  34Ø
C      NBREAK     KBREAK         Ø                1 2 3                PPF  35Ø
C      XBREAK     BRAKPT         -                                     PPF  36Ø
C      DBREAK     KDERVB         -                                     PPF  37Ø
C      BLEFT      VALLFT         -                                     PPF  38Ø
C      BRIGHT     VALRGT         -                                     PPF  39Ø
C                                                                       PPF  4ØØ
      REAL A, ACCUR, B, BLEFT, BRIGHT, CHARF, DDTEMP, DSCTOL, ERROR,    PPF  41Ø
     * ERRORI, FACTOR, FINTRP, FLEFT, FRIGHT, NORM, XBREAK, XDD,        PPF  42Ø
     * XINTRP, XLEFT, XRIGHT                                            PPF  43Ø
      DIMENSION XBREAK(2Ø), DBREAK(2Ø), BLEFT(2Ø), BRIGHT(2Ø)           PPF  44Ø
      DIMENSION XLEFT(5Ø), XRIGHT(5Ø), FACTOR(12), FMESGE(6)            PPF  45Ø
      DIMENSION DDTEMP(12,12), FINTRP(1Ø), FLEFT(6), FRIGHT(6),         PPF  46Ø
     * XDD(12), XINTRP(1Ø)                                              PPF  47Ø
      INTEGER BOTH, BREAK, DBREAK, DEGREE, EDIST, FMESGE, RIGHT,        PPF  48Ø
     * RIGHTX, SMOOTH                                                   PPF  49Ø
      LOGICAL DISCRD, FATAL, FINISH                                     PPF  5ØØ
      REAL XKNOTS(KDIMEN), COEFS(KDIMEN,NDIMEN)                         PPF  51Ø
      REAL BRAKPT, EMEAS, EPSLN, ERREST, FOSCIL, F, VALLFT, VALRGT,     PPF  52Ø
     * XLFT, XRGT                                                       PPF  53Ø
      DIMENSION BRAKPT(2Ø), KDERVB(2Ø), VALLFT(2Ø), VALRGT(2Ø)          PPF  54Ø
      INTEGER ATYPE                                                     PPF  55Ø
      EXTERNAL F                                                        PPF  56Ø
```

```
      COMMON /INPUTZ/ A, B, ACCUR, NORM, CHARF, XBREAK, BLEFT, BRIGHT,  PPF  57Ø
     * DBREAK, DEGREE, SMOOTH, LEVEL, EDIST, NBREAK, KNTDIM, NPARDM     PPF  58Ø
      COMMON /RESULZ/ ERROR, KNOTS                                      PPF  59Ø
      COMMON /KONTRL/ DSCTOL, ERRORI, XLEFT, XRIGHT, BREAK, BOTH,       PPF  6ØØ
     * FACTOR, FMESGE, IBREAK, INTERP, LEFT, MAXAUX, MAXKNT, MAXPAR,    PPF  61Ø
     * MAXSTK, NPAR, NSTACK, RIGHT, DISCRD, FATAL, FINISH               PPF  62Ø
      COMMON /COMDIF/ DDTEMP, FINTRP, FLEFT, FRIGHT, XDD, XINTRP,       PPF  63Ø
     * LEFTX, NINTRP, RIGHTX                                            PPF  64Ø
      COMMON /SAVEIT/ IKNOT                                             PPF  65Ø
C                                                                       PPF  66Ø
C                                                                       PPF  67Ø
C             SKIP ALL DEFAULT SETTING FOR PPFIT4                       PPF  68Ø
      GO TO 1Ø                                                          PPF  69Ø
C                                                                       PPF  7ØØ
C                  SET DEFAULTS FOR PPFIT1                              PPF  71Ø
C                                                                       PPF  72Ø
C         ****** SINCE ENTRY IS NON-STANDARD  ******                    PPF  73Ø
C         ****** THE NATURAL WAY TO IMPLEMENT ******                    PPF  74Ø
C         ****** THE OTHER PPFITS IS ONLY     ******                    PPF  75Ø
C         ****** INDICATED IN THE COMMENTS    ******                    PPF  76Ø
C                                                                       PPF  77Ø
C     ENTRY PPFIT1                                                      PPF  78Ø
C     ENTRY PPFIT1(F,XLFT,XRGT,EPSLN,NPIECE,ERREST,XKNOTS,COEFS,        PPF  79Ø
C    A             KDIMEN,NDIMEN)                                     PPF  8ØØ
      NDEG = 3                                                          PPF  81Ø
      NSMTH = Ø                                                         PPF  82Ø
C                                                                       PPF  83Ø
C                  SET DEFAULTS FOR PPFIT2                              PPF  84Ø
C     ENTRY PPFIT2                                                      PPF  85Ø
C     ENTRY PPFIT2(F,XLFT,XRGT,EPSLN,NPIECE,ERREST,XKNOTS,COEFS,        PPF  86Ø
C    A             KDIMEN,NDIMEN,NDEG,NSMTH)                          PPF  87Ø
      EMEAS = 3.Ø                                                       PPF  88Ø
      LPRNT = 1                                                         PPF  89Ø
      FOSCIL = B - A                                                    PPF  9ØØ
      IF (NDEG.EQ.2) FOSCIL = .5*FOSCIL                                 PPF  91Ø
      IF (NDEG.LE.1) FOSCIL = (B-A)/3.                                  PPF  92Ø
C                                                                       PPF  93Ø
C                  SET DEFAULTS FOR PPFIT3                              PPF  94Ø
C     ENTRY PPFIT3                                                      PPF  95Ø
C     ENTRY PPFIT3(F,XLFT,XRGT,EPSLN,NPIECE,ERREST,XKNOTS,COEFS,        PPF  96Ø
C    A             KDIMEN,NDIMEN,NDEG,NSMTH,EMEAS,LPRNT,FOSCIL)       PPF  97Ø
      ATYPE = 1                                                         PPF  98Ø
      KSING = Ø                                                         PPF  99Ø
      KBREAK = Ø                                                        PPF 1ØØØ
C                                                                       PPF 1Ø1Ø
C          PUT INPUT INTO COMMON INPUTZ BY CHANGING TO INTERNAL NAMES    PPF 1Ø2Ø
   1Ø A = XLFT                                                          PPF 1Ø3Ø
      B = XRGT                                                          PPF 1Ø4Ø
      ACCUR = EPSLN                                                     PPF 1Ø5Ø
      DEGREE = NDEG                                                     PPF 1Ø6Ø
      SMOOTH = NSMTH                                                    PPF 1Ø7Ø
      NORM = EMEAS                                                      PPF 1Ø8Ø
      LEVEL = LPRNT                                                     PPF 1Ø9Ø
      CHARF = FOSCIL                                                    PPF 11ØØ
      EDIST = ATYPE                                                     PPF 111Ø
      NBREAK = KBREAK                                                   PPF 112Ø
      IF (NBREAK.LE.Ø .OR. NBREAK.GE.21) GO TO 3Ø                       PPF 113Ø
      DO 2Ø K=1,NBREAK                                                  PPF 114Ø
        XBREAK(K) = BRAKPT(K)                                           PPF 115Ø
        DBREAK(K) = KDERVB(K)                                           PPF 116Ø
        BLEFT(K) = VALLFT(K)                                            PPF 117Ø
        BRIGHT(K) = VALRGT(K)                                           PPF 118Ø
   2Ø CONTINUE                                                          PPF 119Ø
   3Ø CONTINUE                                                          PPF 12ØØ
C                                                                       PPF 121Ø
      CALL ADAPT(F, XKNOTS, COEFS, KDIMEN, NDIMEN)                      PPF 122Ø
      NPIECE = KNOTS                                                    PPF 123Ø
      ERREST = ERROR                                                    PPF 124Ø
      RETURN                                                            PPF 125Ø
      END                                                               PPF 126Ø



      SUBROUTINE CHECK(FUNCT, CHAR)                                     CHE   1Ø
C                                                                       CHE   2Ø
```

```
C       ================================================================CHE   3Ø
C                                                                       CHE   4Ø
C **  THIS PROGRAM CHECKS FOR DISCARDING INTERVAL, APPLIES VARIOUS      CHE   5Ø
C     TESTS ABOUT DISCARDING INVOLVING EDIST AND CHARF.                 CHE   6Ø
C                                                                       CHE   7Ø
      REAL A, ACCUR, B, BLEFT, BRIGHT, CHARF, DDTEMP, DSCTOL, ERROR,    CHE   8Ø
     * ERRORI, FACTOR, FINTRP, FLEFT, FRIGHT, NORM, XBREAK, XDD,        CHE   9Ø
     * XINTRP, XLEFT, XRIGHT                                           CHE  1ØØ
      DIMENSION XBREAK(2Ø), DBREAK(2Ø), BLEFT(2Ø), BRIGHT(2Ø)           CHE  11Ø
      DIMENSION XLEFT(5Ø), XRIGHT(5Ø), FACTOR(12), FMESGE(6)            CHE  12Ø
      DIMENSION DDTEMP(12,12), FINTRP(1Ø), FLEFT(6), FRIGHT(6),         CHE  13Ø
     * XDD(12), XINTRP(1Ø)                                             CHE  14Ø
      INTEGER BOTH, BREAK, DBREAK, DEGREE, EDIST, FMESGE, RIGHT,        CHE  15Ø
     * RIGHTX, SMOOTH                                                  CHE  16Ø
      LOGICAL DISCRD, FATAL, FINISH                                     CHE  17Ø
      REAL DTEST, DX                                                    CHE  18Ø
      COMMON /INPUTZ/ A, B, ACCUR, NORM, CHARF, XBREAK, BLEFT, BRIGHT,  CHE  19Ø
     * DBREAK, DEGREE, SMOOTH, LEVEL, EDIST, NBREAK, KNTDIM, NPARDM     CHE  2ØØ
      COMMON /RESULZ/ ERROR, KNOTS                                      CHE  21Ø
      COMMON /KONTRL/ DSCTOL, ERRORI, XLEFT, XRIGHT, BREAK, BOTH,       CHE  22Ø
     * FACTOR, FMESGE, IBREAK, INTERP, LEFT, MAXAUX, MAXKNT, MAXPAR,    CHE  23Ø
     * MAXSTK, NPAR, NSTACK, RIGHT, DISCRD, FATAL, FINISH               CHE  24Ø
      COMMON /COMDIF/ DDTEMP, FINTRP, FLEFT, FRIGHT, XDD, XINTRP,       CHE  25Ø
     * LEFTX, NINTRP, RIGHTX                                           CHE  26Ø
C                                                                       CHE  27Ø
C                                                                       CHE  28Ø
      DISCRD = .FALSE.                                                  CHE  29Ø
      DX = XRIGHT(NSTACK) - XLEFT(NSTACK)                               CHE  3ØØ
C                                                                       CHE  31Ø
C         COMPUTE DTEST FOR IMPLEMENTING VARIOUS TYPES OF ADAPTIVE APPRXCHE  32Ø
      IF (EDIST.EQ.Ø) DTEST = DX*DSCTOL                                 CHE  33Ø
C         FOR THE APPROXIMATE FIXED ERROR DISTRIBUTION TYPE WE ESTIMATE CHE  34Ø
C         THE FINAL NUMBER OF KNOTS BY( LIMITING IT A LITTLE )          CHE  35Ø
C             (NSTACK+KNOTS+2)((B-A)/(XRIGHT-A))                        CHE  36Ø
      IF (EDIST.EQ.1) DTEST = DSCTOL/(FLOAT(NSTACK+KNOTS+2)*AMIN1((B-A)/CHE  37Ø
     * (XRIGHT(NSTACK)-A),5.))                                          CHE  38Ø
      IF (EDIST.EQ.2 .OR. NORM.EQ.3.) DTEST = DSCTOL                    CHE  39Ø
C                                                                       CHE  4ØØ
C             CHECK FOR DISCARD OF INTERVAL                             CHE  41Ø
      IF (ERRORI.LE.DTEST) DISCRD = .TRUE.                              CHE  42Ø
C                                                                       CHE  43Ø
C         CHECK CHARACTERISTIC LENGTH OF FUNCTION                       CHE  44Ø
      IF (DX.GE.CHARF) DISCRD = .FALSE.                                 CHE  45Ø
      RETURN                                                            CHE  46Ø
      END                                                               CHE  47Ø



      SUBROUTINE COMPUT(F, APPROX, INTERV)                              COM   1Ø
C                                                                       COM   2Ø
C       ================================================================COM   3Ø
C                                                                       COM   4Ø
C **  THIS PROGRAM COMPUTES THE PIECEWISE POLYNOMIAL APPROXIMATION ON   COM   5Ø
C     THE CURRENT INTERVAL. IT ALSO ESTIMATES THE ERROR                 COM   6Ø
C                                                                       COM   7Ø
      REAL A, ACCUR, B, BLEFT, BRIGHT, CHARF, DDTEMP, DSCTOL, ERROR,    COM   8Ø
     * ERRORI, FACTOR, FINTRP, FLEFT, FRIGHT, NORM, XBREAK, XDD,        COM   9Ø
     * XINTRP, XLEFT, XRIGHT                                           COM  1ØØ
      DIMENSION XBREAK(2Ø), DBREAK(2Ø), BLEFT(2Ø), BRIGHT(2Ø)           COM  11Ø
      DIMENSION XLEFT(5Ø), XRIGHT(5Ø), FACTOR(12), FMESGE(6)            COM  12Ø
      DIMENSION DDTEMP(12,12), FINTRP(1Ø), FLEFT(6), FRIGHT(6),         COM  13Ø
     * XDD(12), XINTRP(1Ø)                                             COM  14Ø
      INTEGER BOTH, BREAK, DBREAK, DEGREE, EDIST, FMESGE, RIGHT,        COM  15Ø
     * RIGHTX, SMOOTH                                                  COM  16Ø
      LOGICAL DISCRD, FATAL, FINISH                                     COM  17Ø
      REAL ABSC, DX, ERRINT, F, FDERVL, FDERVR, FDUMB, POLYDD, WGTS     COM  18Ø
      DIMENSION ABSC(4), WGTS(4), FDERVL(5), FDERVR(5), FDUMB(5)        COM  19Ø
      EXTERNAL F, POLYDD                                                COM  2ØØ
      COMMON /INPUTZ/ A, B, ACCUR, NORM, CHARF, XBREAK, BLEFT, BRIGHT,  COM  21Ø
     * DBREAK, DEGREE, SMOOTH, LEVEL, EDIST, NBREAK, KNTDIM, NPARDM     COM  22Ø
      COMMON /RESULZ/ ERROR, KNOTS                                      COM  23Ø
      COMMON /KONTRL/ DSCTOL, ERRORI, XLEFT, XRIGHT, BREAK, BOTH,       COM  24Ø
     * FACTOR, FMESGE, IBREAK, INTERP, LEFT, MAXAUX, MAXKNT, MAXPAR,    COM  25Ø
     * MAXSTK, NPAR, NSTACK, RIGHT, DISCRD, FATAL, FINISH               COM  26Ø
      COMMON /COMDIF/ DDTEMP, FINTRP, FLEFT, FRIGHT, XDD, XINTRP,       COM  27Ø
     * LEFTX, NINTRP, RIGHTX                                           COM  28Ø
```

```
C                                                                       COM  29Ø
      EQUIVALENCE (FLEFT(2),FDERVL(1)), (FRIGHT(2),FDERVR(1))           COM  3ØØ
C     FIFTEEN DIGIT VALUES FOR THESE GAUSS INTEGRATION CONSTANTS ARE    COM  31Ø
C  .861136311594Ø53 .3399810Ø43584856 .347854845137454 .652145154862546 COM  32Ø
      DATA ABSC(1) /-.86113631159405/                                   COM  33Ø
      DATA ABSC(2) /-.3399810Ø4358486/                                  COM  34Ø
      DATA ABSC(3) /.3399810Ø4358486/                                   COM  35Ø
      DATA ABSC(4) /.86113631159405/                                    COM  36Ø
      DATA WGTS(1) /.34785484513745/                                    COM  37Ø
      DATA WGTS(2) /.65214515486255/                                    COM  38Ø
      DATA WGTS(3) /.65214515486255/                                    COM  39Ø
      DATA WGTS(4) /.34785484513745/                                    COM  4ØØ
C                                                                       COM  41Ø
C             COMPUTE INTERPOLATION INFORMATION                         COM  42Ø
      NINTRP = DEGREE - 2*SMOOTH - 1                                    COM  43Ø
C                                                                       COM  44Ø
C          INCREASE NUMBER OF INTERPOLATION POINTS IF BREAK POINTS ARE   COM  45Ø
C          SPECIFIED WITH FEWER DERIVATIVES THAN SMOOTH                 COM  46Ø
      IF (BREAK.EQ.LEFT .OR. BREAK.EQ.RIGHT) NINTRP = NINTRP + SMOOTH - COM  47Ø
     * DBREAK(IBREAK)                                                   COM  48Ø
      IF (BREAK.EQ.BOTH) NINTRP = NINTRP + 2*SMOOTH - DBREAK(IBREAK) -  COM  49Ø
     * DBREAK(IBREAK+1)                                                COM  5ØØ
      IF (NINTRP.EQ.Ø) GO TO 2Ø                                         COM  51Ø
C             GENERATE EQUAL SPACED INTERPOLATION POINTS                COM  52Ø
      DX = (XRIGHT(NSTACK)-XLEFT(NSTACK))/FLOAT(NINTRP+1)               COM  53Ø
      DO 1Ø J=1,NINTRP                                                  COM  54Ø
        XINTRP(J) = XLEFT(NSTACK) + FLOAT(J)*DX                         COM  55Ø
   1Ø CONTINUE                                                          COM  56Ø
C                                                                       COM  57Ø
C             GET LEFT AND RIGHT F-VALUES, PUT F-VALUE IN FIRST ELEMENT COM  58Ø
C             OF ARRAYS FLEFT AND FRIGHT.  GET DERIVATIVES BACK AS      COM  59Ø
C             OTHER ELEMENTS VIA THE SUBARRAYS FDERVL AND FDERVR.       COM  6ØØ
   2Ø FLEFT(1) = F(XLEFT(NSTACK),FDERVL)                                COM  61Ø
      FRIGHT(1) = F(XRIGHT(NSTACK),FDERVR)                              COM  62Ø
      LEFTX = SMOOTH + 1                                                COM  63Ø
      RIGHTX = LEFTX                                                    COM  64Ø
C           GET F-VALUES AT OTHER INTERPOLATION POINTS, IF ANY          COM  65Ø
      IF (NINTRP.EQ.Ø) GO TO 4Ø                                         COM  66Ø
      DO 3Ø J=1,NINTRP                                                  COM  67Ø
        FINTRP(J) = F(XINTRP(J),FDUMB)                                  COM  68Ø
   3Ø CONTINUE                                                          COM  69Ø
C                                                                       COM  7ØØ
C          CHECK FOR BREAK POINTS, MODIFY VALUES IF NECESSARY           COM  71Ø
   4Ø CONTINUE                                                          COM  72Ø
      IF (BREAK.NE.LEFT) GO TO 5Ø                                       COM  73Ø
      LEFTX = DBREAK(IBREAK) + 1                                        COM  74Ø
      FLEFT(LEFTX) = BRIGHT(IBREAK)                                     COM  75Ø
   5Ø IF (BREAK.NE.RIGHT) GO TO 6Ø                                      COM  76Ø
      RIGHTX = DBREAK(IBREAK) + 1                                       COM  77Ø
      FRIGHT(RIGHTX) = BLEFT(IBREAK)                                    COM  78Ø
   6Ø IF (BREAK.NE.BOTH) GO TO 7Ø                                       COM  79Ø
      LEFTX = DBREAK(IBREAK) + 1                                        COM  8ØØ
      RIGHTX = DBREAK(IBREAK+1) + 1                                     COM  81Ø
      FLEFT(LEFTX) = BRIGHT(IBREAK)                                     COM  82Ø
      FRIGHT(RIGHTX) = BLEFT(IBREAK+1)                                  COM  83Ø
   7Ø CONTINUE                                                          COM  84Ø
C                                                                       COM  85Ø
C           COMPUTE DIVIDED DIFFERENCES, NEWTON FORM OF POLYNOMIAL      COM  86Ø
      CALL NEWTON(LEFTX, RIGHTX, NINTRP)                                COM  87Ø
C                                                                       COM  88Ø
C         COMPUTE NORM OF ERROR OF THIS APPROMIMATION USING FOUR PTS     COM  89Ø
C         ADD 5Ø PERCENT FUDGE FACTOR                                   COM  9ØØ
      ERRORI = ERRINT(F,POLYDD,XLEFT(NSTACK),XRIGHT(NSTACK),ABSC,WGTS)  COM  91Ø
      ERRORI = 1.5*ERRORI                                               COM  92Ø
      RETURN                                                            COM  93Ø
      END                                                               COM  94Ø


      REAL FUNCTION ERRINT(F, FIT, AAA, BBB, POINTS, WEIGHT)            ERR   1Ø
C                                                                       ERR   2Ø
C       ================================================================ERR   3Ø
C                                                                       ERR   4Ø
C **  THIS FUNCTION DOES A FOUR POINT INTEGRATION RULE FOR THE          ERR   5Ø
C     ABSOLUTE VALUE OF THE DIFFERENCE OF TWO FUNCTIONS( F AND FIT )    ERR   6Ø
C                   ABS( F(X) - FIT(X) )**NORM                          ERR   7Ø
C     THE INTEGRATION USES THE POINTS AND WEIGHTS GIVEN AND SCALED      ERR   8Ø
```

```
C     FROM (-1,1) TO (AAA,BBB)                                          ERR   9Ø
C                                                                       ERR   1ØØ


      SUBROUTINE NEWTON(NL, NR, NI)                                     NEW   1Ø
C                                                                       NEW   2Ø
C        ==============================================================NEW   3Ø
C                                                                       NEW   4Ø
C **  THIS PROGRAM COMPUTES THE DIVIDED DIFFERENCES ARRAY AS FOLLOWS     NEW   5Ø
C         NL COALESCED POINTS ON LEFT   - DERIV VALUES IN FLEFT         NEW   6Ø
C         NR COALESCED POINTS ON RIGHT  -   -       -    - FRIGHT       NEW   7Ø
C         NI DISTINCT  POINTS INBETWEEN - FNCTN     -    - FINTRP       NEW   8Ø
C                                                                       NEW   9Ø
C     THE POINTS ARE ORDERED XL = XLEFT (NSTACK)                        NEW   1ØØ
C                            XR = XRIGHT(NSTACK)                        NEW   11Ø
C                            XINTRP ARRAY                               NEW   12Ø
C                                                                       NEW   13Ø
C         LAYOUT OF THE DDTEMP DIVIDED DIFFERENCE ARRAY                 NEW   14Ø
C                                                                       NEW   15Ø
C     NL=6    LLLLLL****II                                              NEW   16Ø
C     NR=4    LLLLL****II      L = FIRST TRIANGLE                       NEW   17Ø
C     NI=2    LLLL****II                                                NEW   18Ø
C             LLL****II        R = SECOND TRIANGLE                      NEW   19Ø
C             LL****II                                                  NEW   2ØØ
C             L****II          * = FILL BETWEEN TRIANGLES               NEW   21Ø
C             RRRRII                                                    NEW   22Ø
C             RRRII            I = COMPLETION FOR INTERPOLATION POINTS  NEW   23Ø
C             RRII                                                      NEW   24Ø
C             RII         IDIF = HORIZONTAL COORD. = DIFFERENCE ORDER   NEW   25Ø
C             II          IPT  = VERTICAL COORD. ASSOCIATED WITH POINTS NEW   26Ø
C             I                                                         NEW   27Ø
C                                                                       NEW   28Ø
C                                                                       NEW   29Ø
      REAL A, ACCUR, B, BLEFT, BRIGHT, CHARF, DDTEMP, DSCTOL, ERROR,    NEW   3ØØ
     * ERRORI, FACTOR, FINTRP, FLEFT, FRIGHT, NORM, XBREAK, XDD,        NEW   31Ø
     * XINTRP, XLEFT, XRIGHT                                           NEW   32Ø
      DIMENSION XBREAK(2Ø), DBREAK(2Ø), BLEFT(2Ø), BRIGHT(2Ø)           NEW   33Ø
      DIMENSION XLEFT(5Ø), XRIGHT(5Ø), FACTOR(12), FMESGE(6)            NEW   34Ø
      DIMENSION DDTEMP(12,12), FINTRP(1Ø), FLEFT(6), FRIGHT(6),         NEW   35Ø
     * XDD(12), XINTRP(1Ø)                                             NEW   36Ø
      INTEGER BOTH, BREAK, DBREAK, DEGREE, EDIST, FMESGE, RIGHT,        NEW   37Ø
     * RIGHTX, SMOOTH                                                  NEW   38Ø
      LOGICAL DISCRD, FATAL, FINISH                                     NEW   39Ø
      REAL DIFFF, DIFFX                                                 NEW   4ØØ
      COMMON /INPUTZ/ A, B, ACCUR, NORM, CHARF, XBREAK, BLEFT, BRIGHT,  NEW   41Ø
     * DBREAK, DEGREE, SMOOTH, LEVEL, EDIST, NBREAK, KNTDIM, NPARDM     NEW   42Ø
      COMMON /RESULZ/ ERROR, KNOTS                                      NEW   43Ø
      COMMON /KONTRL/ DSCTOL, ERRORI, XLEFT, XRIGHT, BREAK, BOTH,       NEW   44Ø
     * FACTOR, FMESGE, IBREAK, INTERP, LEFT, MAXAUX, MAXKNT, MAXPAR,    NEW   45Ø
     * MAXSTK, NPAR, NSTACK, RIGHT, DISCRD, FATAL, FINISH               NEW   46Ø
      COMMON /COMDIF/ DDTEMP, FINTRP, FLEFT, FRIGHT, XDD, XINTRP,       NEW   47Ø
     * LEFTX, NINTRP, RIGHTX                                           NEW   48Ø
C                                                                       NEW   49Ø
C                MAIN CALCULATION OF DIVIDED DIFFERENCES                NEW   5ØØ
C         DEFINE A FEW SHORT CONSTANTS                                  NEW   51Ø
      NL1 = NL - 1                                                      NEW   52Ø
      NL2 = NL + 1                                                      NEW   53Ø
      NR1 = NR - 1                                                      NEW   54Ø
      NR2 = NR + 1                                                      NEW   55Ø
      NRL = NR + NL                                                     NEW   56Ø
C                                                                       NEW   57Ø
C         PUT X-VALUES IN A SINGLE ARRAY WITH NDDX = NL+NR+NI POINTS    NEW   58Ø
      DO 1Ø NDDX=1,NL                                                   NEW   59Ø
        XDD(NDDX) = XLEFT(NSTACK)                                       NEW   6ØØ
   1Ø CONTINUE                                                          NEW   61Ø
      NDDX = NL                                                         NEW   62Ø
      DO 2Ø K=1,NR                                                      NEW   63Ø
        NDDX = NDDX + 1                                                 NEW   64Ø
        XDD(NDDX) = XRIGHT(NSTACK)                                      NEW   65Ø
   2Ø CONTINUE                                                          NEW   66Ø
C            CHECK IF THERE ARE ANY INTERPOLATION POINTS TO ADD TO XDD  NEW   67Ø
      IF (NI.EQ.Ø) GO TO 4Ø                                             NEW   68Ø
      DO 3Ø K=1,NI                                                      NEW   69Ø
        NDDX = NDDX + 1                                                 NEW   7ØØ
        XDD(NDDX) = XINTRP(K)                                           NEW   71Ø
   3Ø CONTINUE                                                          NEW   72Ø
```

```
C                                                                       NEW  73Ø
C           FILL BORDER OF FIRST TRIANGLE - SIZE NL.                    NEW  74Ø
   4Ø CONTINUE                                                          NEW  75Ø
C         TOP BORDER                                                    NEW  76Ø
      DO 5Ø IDIF=1,NL                                                   NEW  77Ø
        DDTEMP(IDIF,1) = FLEFT(IDIF)/FACTOR(IDIF)                       NEW  78Ø
   5Ø CONTINUE                                                          NEW  79Ø
      IF (NL1.EQ.Ø) GO TO 7Ø                                            NEW  8ØØ
C                   BOTTOM BORDER                                       NEW  81Ø
      DO 6Ø IDIF=1,NL1                                                  NEW  82Ø
        IPT = NL2 - IDIF                                                NEW  83Ø
        DDTEMP(IDIF,IPT) = DDTEMP(IDIF,1)                               NEW  84Ø
   6Ø CONTINUE                                                          NEW  85Ø
C                                                                       NEW  86Ø
C           FILL BORDER OF SECOND TRIANGLE - SIZE NR                    NEW  87Ø
   7Ø CONTINUE                                                          NEW  88Ø
C         TOP BORDER                                                    NEW  89Ø
      DO 8Ø IDIF=1,NR                                                   NEW  9ØØ
        DDTEMP(IDIF,NL2) = FRIGHT(IDIF)/FACTOR(IDIF)                    NEW  91Ø
   8Ø CONTINUE                                                          NEW  92Ø
      IF (NRL.EQ.NL2) GO TO 1ØØ                                         NEW  93Ø
C                   BOTTOM BORDER                                       NEW  94Ø
      DO 9Ø IDIF=1,NR1                                                  NEW  95Ø
        IPT = NRL + 1 - IDIF                                            NEW  96Ø
        DDTEMP(IDIF,IPT) = DDTEMP(IDIF,NL2)                             NEW  97Ø
   9Ø CONTINUE                                                          NEW  98Ø
C                                                                       NEW  99Ø
C           FILL PARALLOGRAM BETWEEN THE TWO TRIANGLES JUST FILLED      NEW 1ØØØ
C           FILL ENTRIES PARALLEL TO BOTTOM OF FIRST TRIANGLE           NEW 1Ø1Ø
  1ØØ CONTINUE                                                          NEW 1Ø2Ø
C                                                                       NEW 1Ø3Ø
C         LOOP STEPPING ALONG TOP SIDE OF SECOND TRIANGLE               NEW 1Ø4Ø
      DO 12Ø J=2,NR2                                                    NEW 1Ø5Ø
        IDIF = J                                                        NEW 1Ø6Ø
C             LOOP STEPPING PARALLEL TO BOTTOM SIDE OF FIRST TRIANGLE   NEW 1Ø7Ø
        DO 11Ø K=2,NL2                                                  NEW 1Ø8Ø
          IPT = NL + 2 - K                                              NEW 1Ø9Ø
          DIFFF = DDTEMP(IDIF-1,IPT+1) - DDTEMP(IDIF-1,IPT)             NEW 11ØØ
          IPT2 = IPT - 1 + IDIF                                         NEW 111Ø
          DIFFX = XDD(IPT2) - XDD(IPT)                                  NEW 112Ø
          DDTEMP(IDIF,IPT) = DIFFF/DIFFX                                NEW 113Ø
          IDIF = IDIF + 1                                               NEW 114Ø
  11Ø   CONTINUE                                                        NEW 115Ø
  12Ø CONTINUE                                                          NEW 116Ø
C             DEBUG  DEBUG  DEBUG  DEBUG                                NEW 117Ø
      IF (LEVEL.GE.4 .AND. KNOTS.LE.1) WRITE (6,99999) NR2, NL2, IDIF,  NEW 118Ø
     * IPT, DIFFF, DIFFX                                                NEW 119Ø
C                                                                       NEW 12ØØ
C         FILL IN BOTTOM DIAGONALS FOR INTERPOLATION POINTS, IF ANY     NEW 121Ø
      IF (NI.EQ.Ø) GO TO 15Ø                                            NEW 122Ø
C         LOOP THROUGH THE INTERPOLATATION POINTS                       NEW 123Ø
      DO 14Ø J=1,NI                                                     NEW 124Ø
        IDIF = 2                                                        NEW 125Ø
        NRLJ = NRL + J                                                  NEW 126Ø
        DDTEMP(1,NRLJ) = FINTRP(J)                                      NEW 127Ø
C         LOOP THROUGH THE DIFFERENCES (IDIF INDEX)                     NEW 128Ø
        NRLJ1 = NRLJ - 1                                                NEW 129Ø
        DO 13Ø K=1,NRLJ1                                                NEW 13ØØ
          IPT = NRLJ - K                                                NEW 131Ø
          DIFFF = DDTEMP(IDIF-1,IPT+1) - DDTEMP(IDIF-1,IPT)             NEW 132Ø
          DIFFX = XDD(NRLJ) - XDD(IPT)                                  NEW 133Ø
          DDTEMP(IDIF,IPT) = DIFFF/DIFFX                                NEW 134Ø
          IDIF = IDIF + 1                                               NEW 135Ø
  13Ø   CONTINUE                                                        NEW 136Ø
  14Ø CONTINUE                                                          NEW 137Ø
  15Ø CONTINUE                                                          NEW 138Ø
      RETURN                                                            NEW 139Ø
99999 FORMAT (9X, 3ØHNR2,NL2,IDIF,IPT,DIFFF,DIFFX =, 4I3, 2F12.6)       NEW 14ØØ
      END                                                               NEW 141Ø


      REAL FUNCTION POLYDD(X)                                           POL   1Ø
C                                                                       POL   2Ø
C       ================================================================POL   3Ø
C                                                                       POL   4Ø
C **  THIS FUNCTION EVALUATES THE CURRENT POLYNOMIAL PIECE REPRESENTED   POL   5Ø
```

```
C     BY THE DIVIDED DIFFERENCES DDTEMP ON THE POINT SET XDD.          POL   6Ø
C                                                                      POL   7Ø

      REAL FUNCTION PPOLY(T, XKNOTS, COEFS, KDIMEN, NDIMEN)            PPO   1Ø
C                                                                      PPO   2Ø
C        ==============================================================PPO   3Ø
C                                                                      PPO   4Ø
C **  THIS FUNCTION EVALUATES THE PIECEWISE POLYNOMIAL APPROXIMATION    PPO   5Ø
C     COMPUTED IN ADAPT                                                PPO   6Ø
C                                                                      PPO   7Ø

      SUBROUTINE PTRANS(D, POWERS)                                     PTR   1Ø
C                                                                      PTR   2Ø
C        ==============================================================PTR   3Ø
C                                                                      PTR   4Ø
C **  THIS PROGRAM CONVERTS POLYNOMIAL REPRESENTATION FROM DIVIDED      PTR   5Ø
C     DIFFERENCE TO POWER FORM.  THERE ARE COALESCED POINTS ON EACH    PTR   6Ø
C     END OF THE INTERVAL (XL,XR) = (XLEFT(NSTACK),XRIGHT(NSTACK)).    PTR   7Ø
C     THE NUMBER COALESCED AT EACH END IS LEFTX AND RIGHTX.            PTR   8Ø
C     AND THERE ARE NINTRP OTHER PTS XINTRP(K)  INBETWEEN THEM.        PTR   9Ø
C     SEE SUBROUTINE NEWTON FOR MORE DETAILS                           PTR  1ØØ
C                                                                      PTR  11Ø

C          STARTING REPRESENTATION IS (ASSUMING XL = Ø )                PTR  45Ø
C                                                                       PTR  46Ø
C    D(1) +D(2)X +D(3)X**2 + --- +D(NL)X**(NL-1)                        PTR  47Ø
C   +(X**NL)*( D(NL+1)(+D(NL+2)(X-XR)**2 + --- +D(NL+NR)*(X-XR)**(NR-1) PTR  48Ø
C        *((X-XR)**NR)*(D(NL+NR+1) + D(NL+NR+2)*(X-XINTRP(1))           PTR  49Ø
C                      +D(NL+NR+3)*(X-XINTRP(1))(X-XINTRP(2)) + ---))   PTR  5ØØ
C                                                                       PTR  51Ø
C          STRATEGY IS TO FIRST CONVERT THE PART FROM THE INTERP. PTS.  PTR  52Ø
C          TO POLY IN (X-XR).  THIS POLY THEN HAS ORIGIN SHIFTED TO XL. PTR  53Ø
C                                                                       PTR  54Ø
C     THE CONVERSION OF THE INTERP PART IS DONE EXPLICITLY FOR DEGREE   PTR  55Ø
C     TWO OR LESS AND DONE BY SYNTHETIC DIVISION FOR HIGHER DEGREES     PTR  56Ø
C                                                                       PTR  57Ø
C   D1 + D2(X-X1) +D3(X**2-(X1+X2)X +X1*X2)                             PTR  58Ø
C                                                                       PTR  59Ø
C     THE RESULTING COEFFICIENTS ARE PUT IN THE ARRAY POWERS            PTR  6ØØ
C                                                                       PTR  61Ø

      SUBROUTINE PUT(INTERV, XKNOTS, COEFS, KDIMEN, NDIMEN)            PUT   1Ø
C                                                                      PUT   2Ø
C        ==============================================================PUT   3Ø
C **  THIS PROGRAM PUTS INTERVALS ON THE STACK OR DISCARDS THEM.        PUT   4Ø
C     WHEN AN INTERVAL IS DISCARDED A NEW KNOT IS FOUND. THEN THIS     PUT   5Ø
C     PROGRAM UPDATES THE ERROR ESTIMATE, THE XKNOT ARRAY, TRANSFORMS  PUT   6Ø
C     THE POLYNOMIAL TO THE POWER FORM AND PUT THE COEFFICIENTS INTO    PUT   7Ø
C     THE ARRAY COEFS.  IT ALSO CHECKS FOR PASSING BREAK POINTS        PUT   8Ø
C                                                                      PUT   9Ø

      SUBROUTINE SETUP(XKNOTS, COEFS, KDIMEN, NDIMEN)                  SET   1Ø
C                                                                      SET   2Ø
C        ==============================================================SET   3Ø
C                                                                      SET   4Ø
C **  THIS PROGRAM CHECKS THE INPUT DATA AND INITIALIZES THE COMPUTATIONSET   5Ø
C                                                                      SET   6Ø

      SUBROUTINE SUMARY(XKNOTS, COEFS, KDIMEN, NDIMEN)                 SUM   1Ø
C                                                                      SUM   2Ø
C        ==============================================================SUM   3Ø
C                                                                      SUM   4Ø
C **  THIS PROGRAM PRINTS OUT A SUMMARY OF RESULTS OF ADAPT            SUM   5Ø
C                                                                      SUM   6Ø

      SUBROUTINE TAKE(INTERV)                                          TAK   1Ø
C                                                                      TAK   2Ø
C        ==============================================================TAK   3Ø
C                                                                      TAK   4Ø
C **  THIS PROGRAM TAKES AN ACTIVE INTERVAL OFF THE TOP OF THE STACK   TAK   5Ø
```

```
C     IT ALSO DOES MOST OF THE WORK OF LOCATING AND HANDLING            TAK   6Ø
C     BREAK POINTS                                                      TAK   7Ø
C                                                                       TAK   8Ø


      SUBROUTINE TERMIN(TEST, AND, PRINT, XKNOTS, KDIMEN)               TER   1Ø
C                                                                       TER   2Ø
C       ================================================================TER   3Ø
C                                                                       TER   4Ø
C **  THIS PROGRAM TESTS FOR TERMINATION AND GIVES INTERMEDIATE OUTPUT   TER   5Ø
C                                                                       TER   6Ø


      BLOCK DATA                                                        BLK   1Ø
C         SET PARAMETERS FOR FUNCTIONS IN F                             BLK   2Ø
      REAL PAR2, PAR3A, PAR3B, PAR4, PAR4A, P5A, P5B, P6A, P6B, P7,     BLK   3Ø
     * P8A, P8B, P8C, P8D, P8E, P8F, P8G, P8H, P9, P1ØA, P1ØB, P1ØC     BLK   4Ø
      COMMON /FPARS/ PAR2, PAR3A, PAR3B, PAR4, PAR4A, P5A, P5B, P6A,    BLK   5Ø
     * P6B, P7, P8A, P8B, P8C, P8D, P8E, P8F, P8G, P8H, P9, P1ØA, P1ØB, BLK   6Ø
     * P1ØC                                                            BLK   7Ø
      DATA PAR2, PAR3A, PAR3B, PAR4, PAR4A, P5A, P5B                    BLK   8Ø
     * /.Ø2,.4,.6,.ØØ1,6.,8ØØØ.,2Ø37./                                 BLK   9Ø
      DATA P6A, P6B, P7, P8A, P8B, P8C, P8D, P8E /4ØØØ.,3998.,.ØØØ1,-2.,BLK  1ØØ
     * 1.5,7.,5.,9.6/                                                  BLK  11Ø
      DATA P8F, P8G, P8H, P9, P1ØA, P1ØB, P1ØC /.6,14.,16.,1.Ø,.25,.5, BLK  12Ø
     * .6667/                                                          BLK  13Ø
      END                                                               BLK  14Ø

C                                                                       MAN,  1Ø
C  **** THIS PROGRAM RUNS ADAPT WITH DATA READ IN FROM CARDS            MAN   2Ø
C       IT IS DESIGNED TO TEST ADAPT WITH THE 2Ø FUNCTIONS IN F(X).     MAN   3Ø
C       IT READS DATA FOR EVERYTHING EXCEPT EDIST(=ATYPE) AND PRINTS    MAN   4Ø
C       A HEADING WITH THE FUNCTIONS NAME. IT CALLS AND TIMES ADAPT,    MAN   5Ø
C       THEN INDEPENDENTLY CHECKS THE ERROR IN ADAPT AND PLOTS THE      MAN   6Ø
C       FUNCTION AND ERROR CURVE.                                       MAN   7Ø
C                                                                       MAN   8Ø
C NOTES -- USER MUST SUPPLY SYSTEM ROUTINES SECOND AND GRAPH            MAN   9Ø
C          INTEGRATION USES ERRINT FROM ADAPT.                          MAN  1ØØ
C          PRESENT DIMENSIONS (ON ZKNOTS + ZCOEFS) PREVENT USING        MAN  11Ø
C             ERRINT TO CHECK ACCURACY FOR MORE THAN 1ØØ KNOTS OR       MAN  12Ø
C             POLYNOMIAL DEGREE 6.                                      MAN  13Ø
C          QPOLY USED BY ERRINT IS A SINGLE ARGUMENT VERSION OF PPOLY.  MAN  14Ø
C          SEE COMMENTS IN ADAPT ABOUT DIMENSION CONSTRAINTS AND        MAN  15Ø
C             PORTABILITY.                                              MAN  16Ø
C          IF ADAPT AND THIS DRIVER ARE CHANGED TO DOUBLE PRECISION     MAN  17Ø
C             ONE PROBABLY WANTS TO LEAVE TIME,TSTART,TSTOP,XVALU,      MAN  18Ø
C             GRAF1,GRAF2 AS SINGLE PRECISION                           MAN  19Ø
C             THEY ARE SEPARATELY DECLARED HERE.                        MAN  2ØØ
C                                                                       MAN  21Ø


      REAL FUNCTION F(X, FDERV)                                         F     1Ø
C                                                                       F     2Ø
C **  AN ARRAY OF TWENTY TEST FUNCTIONS WITH 2 TO 5 DERIVATIVES.        F     3Ø
C     THE DERIVATIVE VALUES ARE PLACED IN FDERV - DIMENSIONED AT 5      F     4Ø
C     THE FUNCTIONS OF THE SECOND GROUP ARE PARAMETERIZED IN VARIOUS    F     5Ø
C     WAYS.  THE PARAMETERS ARE SET IN BLOCK DATA AND AVAILABLE         F     6Ø
C     THROUGH THE COMMON BLOCK / FPARS /                                F     7Ø
C         REQUIRED INPUT INFORMATION IN COMMON BLOCK / FDATA /          F     8Ø
C            JFUNK  = INDEX OF THE FUNCTION SELECTED                    F     9Ø
C            KOUNT  = NUMBER OF F-EVALUATIONS                           F    1ØØ
C            MAXK   = MAXIMUN ALLOWED VALUE OF KOUNT                    F    11Ø
C         REQUIRED CONTROL INFORMATION IN COMMON BLOCK / KONTROL /      F    12Ø
C            FINISH = SWITCH THAT STOPS ADAPT                           F    13Ø
C                                                                       F    14Ø
```

ALGORITHM 526
Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points [E1]

HIROSHI AKIMA
Institute for Telecommunication Sciences

Key Words and Phrases: bivariate interpolation, interpolation, partial derivative, polynomial, smooth surface fitting
CR Categories: 5.13
Language: ANSI Standard Fortran

DESCRIPTION

This algorithm describes the IDBVIP/IDSFFT subprogram package that implements the method of bivariate interpolation and smooth surface fitting for irregularly distributed data points [1]. The package is written in ANSI Standard Fortran [2].

The package consists of nine subprograms, i.e. eight subroutines and a function. Two subroutines, IDBVIP and IDSFFT, are the master subroutines of the package, and each interfaces with the user. The IDBVIP subroutine performs bivariate interpolation; it estimates the z values at the specified points in the x-y plane. The IDSFFT subroutine performs smooth surface fitting; it estimates the z values at the specified rectangular grid points in the x-y plane and generates a doubly-dimensioned array containing these estimated values.

The remaining six subroutines are supporting subroutines called by either IDBVIP or IDSFFT or by both: IDCLDP determines several data points closest to each of the data points; IDGRID organizes output grid points for IDSFFT by sorting them in ascending order of triangle numbers; IDLCTN locates a point (i.e. determines a triangle in which the point lies) for IDBVIP; IDPDRV estimates partial derivatives at the data points; IDPTIP performs punctual interpolation (i.e. interpolation at a point); and IDTANG triangulates (or divides into a number of triangles) the x-y plane. A function, IDXCHG, is called by IDTANG and determines whether or not an exchange of triangles is necessary.

The package includes two common blocks, IDLC and IDPI. Including these common areas, the package occupies approximately 3500 locations on the CDC-6600 computer.

When the user wishes to call either the IDBVIP or the IDSFFT subroutine repeatedly with identical data as parts of input data in two consecutive calls, he can save computation times considerably by specifying an appropriate mode of computation. (This mode is specified with the MD parameter in the call statements.)

Tables I and II show the approximate computation times required for the IDBVIP and IDSFFT subroutines, respectively, on the CDC-6600 computer when

Received 28 December 1976; revised 6 June 1977.
Authors' address: U.S. Department of Commerce, NTIA, Institute for Telecommunication Sciences, Boulder, CO 80303.

TABLE I. Approximate Computation Times Required for the IDBVIP Subroutine on the CDC-6600 Computer

NDP	NIP	Time (seconds)		
		MD = 1	MD = 2	MD = 3
10	10	.03	.02	.02
	100	.13	.12	.08
	1000	1.1	1.1	.70
100	10	.75	.10	.07
	100	.86	.22	.14
	1000	2.1	1.5	.84
1000	10	47.	.84	.60
	100	47.	1.1	.68
	1000	50.	3.8	1.4

TABLE II. Approximate Computation Times Required for the IDSFFT Subroutine on the CDC-6600 Computer

NDP	NXI * NYI	Time (seconds)		
		MD = 1	MD = 2	MD = 3
10	11 * 11	.08	.07	.04
	33 * 33	.41	.40	.22
	101 * 101	3.5	3.4	1.8
100	11 * 11	.86	.20	.14
	33 * 33	1.3	.66	.37
	101 * 101	4.6	4.0	2.0
1000	11 * 11	47.	1.1	.70
	33 * 33	49.	2.6	1.3
	101 * 101	54.	8.2	3.5

four additional data points are used for estimating partial derivatives at each data point (i.e. NCP = 4). In these tables, NDP is the number of data points, NIP is the number of output points at which interpolation is to be performed, and NXI and NYI are the numbers of output grid points in the x and y coordinates. MD is the mode of computation; MD = 1 for a new NCP value and/or a new set of x-y coordinates of data points, MD = 2 for an old NCP value, an old set of x-y coordinates of data points, and a new set of x-y coordinates of output points, and MD = 3 for an old NCP value, an old set of x-y coordinates of data points, and an old set of x-y coordinates of output points.

REFERENCES

1. Akima, H. A method of bivariate interpolation and smooth surface fitting for irregularly distributed data points. *ACM Trans. Math. Software 4*, 2 (June 1978), 148–159.
2. ANSI Standard Fortran. Pub. X3.9-1966. Amer. Nat. Standards Inst., New York; also reproduced in W.P. Heising's history and summary of FORTRAN standardization development for the ASA. *Comm. ACM 7*, 10 (Oct. 1964), 590–625.

ALGORITHM

```
C     PROGRAM  TTIDBS(OUTPUT,TAPE6=OUTPUT)                                  IDØØØØ7Ø
C THIS PROGRAM IS A TEST PROGRAM FOR THE IDBVIP/IDSFFT SUBPRO-              IDØØØØ8Ø
C GRAM PACKAGE.  ALL ELEMENTS OF RESULTING DZI1 AND DZI2 ARRAYS             IDØØØØ9Ø
C ARE EXPECTED TO BE ZERO.                                                  IDØØØ1ØØ
C THE LUN CONSTANT IN THE LAST DATA INITIALIZATION STATEMENT IS             IDØØØ11Ø
C THE LOGICAL UNIT NUMBER OF THE STANDARD OUTPUT UNIT AND IS,               IDØØØ12Ø
C THEREFORE, SYSTEM DEPENDENT.                                              IDØØØ13Ø
C DECLARATION STATEMENTS                                                    IDØØØ14Ø
      DIMENSION   XD(3Ø),YD(3Ø),ZD(3Ø),                                     IDØØØ15Ø
     1            XI(6),YI(5),ZI(6,5),                                     IDØØØ16Ø
     2            ZI1(6,5),ZI2(6,5),DZI1(6,5),DZI2(6,5),                   IDØØØ17Ø
     3            IWK(1Ø3Ø),WK(24Ø)                                        IDØØØ18Ø
      DATA  NCP/4/                                                          IDØØØ19Ø
      DATA  NDP/3Ø/                                                         IDØØØ2ØØ
      DATA  XD(1), XD(2), XD(3), XD(4), XD(5), XD(6),                       IDØØØ21Ø
     1      XD(7), XD(8), XD(9), XD(1Ø),XD(11),XD(12),                     IDØØØ22Ø
     2      XD(13),XD(14),XD(15),XD(16),XD(17),XD(18),                     IDØØØ23Ø
     3      XD(19),XD(2Ø),XD(21),XD(22),XD(23),XD(24),                     IDØØØ24Ø
     4      XD(25),XD(26),XD(27),XD(28),XD(29),XD(3Ø)/                     IDØØØ25Ø
     5      11.16, 24.2Ø, 19.85, 1Ø.35, 19.72,  Ø.ØØ,                      IDØØØ26Ø
     6      2Ø.87, 19.99, 1Ø.28,  4.51,  Ø.ØØ, 16.7Ø,                      IDØØØ27Ø
     7       6.Ø8, 25.ØØ, 14.9Ø,  Ø.ØØ,  9.66,  5.22,                      IDØØØ28Ø
     8      11.77, 15.1Ø, 25.ØØ, 25.ØØ, 14.59, 15.2Ø,                      IDØØØ29Ø
     9       5.23,  2.14,  Ø.51, 25.ØØ, 21.67,  3.31/                      IDØØØ3ØØ
      DATA  YD(1), YD(2), YD(3), YD(4), YD(5), YD(6),                       IDØØØ31Ø
     1      YD(7), YD(8), YD(9), YD(1Ø),YD(11),YD(12),                     IDØØØ32Ø
     2      YD(13),YD(14),YD(15),YD(16),YD(17),YD(18),                     IDØØØ33Ø
     3      YD(19),YD(2Ø),YD(21),YD(22),YD(23),YD(24),                     IDØØØ34Ø
     4      YD(25),YD(26),YD(27),YD(28),YD(29),YD(3Ø)/                     IDØØØ35Ø
     5       1.24, 16.23, 1Ø.72,  4.11,  1.39, 2Ø.ØØ,                      IDØØØ36Ø
     6      2Ø.ØØ,  4.62, 15.16, 2Ø.ØØ,  4.48, 19.65,                      IDØØØ37Ø
     7       4.58, 11.87,  3.12,  Ø.ØØ, 2Ø.ØØ, 14.66,                      IDØØØ38Ø
     8      1Ø.47, 17.19,  3.87,  Ø.ØØ,  8.71,  Ø.ØØ,                      IDØØØ39Ø
     9      1Ø.72, 15.Ø3,  8.37, 2Ø.ØØ, 14.36,  Ø.13/                      IDØØØ4ØØ
      DATA  ZD(1), ZD(2), ZD(3), ZD(4), ZD(5), ZD(6),                       IDØØØ41Ø
     1      ZD(7), ZD(8), ZD(9), ZD(1Ø),ZD(11),ZD(12),                     IDØØØ42Ø
     2      ZD(13),ZD(14),ZD(15),ZD(16),ZD(17),ZD(18),                     IDØØØ43Ø
     3      ZD(19),ZD(2Ø),ZD(21),ZD(22),ZD(23),ZD(24),                     IDØØØ44Ø
     4      ZD(25),ZD(26),ZD(27),ZD(28),ZD(29),ZD(3Ø)/                     IDØØØ45Ø
     5      22.15,  2.83,  7.97, 22.33, 16.83, 34.6Ø,                      IDØØØ46Ø
     6       5.74, 14.72, 21.59, 15.61, 61.77,  6.31,                      IDØØØ47Ø
     7      35.74,  4.4Ø, 21.7Ø, 58.2Ø,  4.73, 4Ø.36,                      IDØØØ48Ø
     8      13.62, 12.57,  8.74, 12.ØØ, 14.81, 21.6Ø,                      IDØØØ49Ø
     9      26.5Ø, 53.1Ø, 49.43,  Ø.6Ø,  5.52, 44.Ø8/                      IDØØØ5ØØ
      DATA  NXI/6/, NYI/5/                                                  IDØØØ51Ø
      DATA  XI(1), XI(2), XI(3), XI(4), XI(5), XI(6)/                       IDØØØ52Ø
     1       Ø.ØØ,  5.ØØ, 1Ø.ØØ, 15.ØØ, 2Ø.ØØ, 25.ØØ/                    IDØØØ53Ø
      DATA  YI(1), YI(2), YI(3), YI(4), YI(5)/                              IDØØØ54Ø
     1       Ø.ØØ,  5.ØØ, 1Ø.ØØ, 15.ØØ, 2Ø.ØØ/                           IDØØØ55Ø
      DATA  ZI(1,1),ZI(2,1),ZI(3,1),ZI(4,1),ZI(5,1),ZI(6,1),               IDØØØ56Ø
     1      ZI(1,2),ZI(2,2),ZI(3,2),ZI(4,2),ZI(5,2),ZI(6,2),               IDØØØ57Ø
     2      ZI(1,3),ZI(2,3),ZI(3,3),ZI(4,3),ZI(5,3),ZI(6,3),               IDØØØ58Ø
     3      ZI(1,4),ZI(2,4),ZI(3,4),ZI(4,4),ZI(5,4),ZI(6,4),               IDØØØ59Ø
     4      ZI(1,5),ZI(2,5),ZI(3,5),ZI(4,5),ZI(5,5),ZI(6,5)/               IDØØØ6ØØ
     5      58.2Ø, 39.55, 26.9Ø, 21.71, 17.68, 12.ØØ,                      IDØØØ61Ø
     6      61.58, 39.39, 22.Ø4, 21.29, 14.36,  8.Ø4,                      IDØØØ62Ø
     7      59.18, 27.39, 16.78, 13.25,  8.59,  5.36,                      IDØØØ63Ø
     8      52.82, 4Ø.27, 22.76, 16.61,  7.4Ø,  2.88,                      IDØØØ64Ø
     9      34.6Ø, 14.Ø5,  4.12,  3.17,  6.31,  Ø.6Ø/                      IDØØØ65Ø
      DATA  LUN/6/                                                          IDØØØ66Ø
C CALCULATION                                                               IDØØØ67Ø
   1Ø MD=1                                                                  IDØØØ68Ø
      DO 12  IYI=1,NYI                                                      IDØØØ69Ø
        DO 11  IXI=1,NXI                                                    IDØØØ7ØØ
          IF(IXI.NE.1.OR.IYI.NE.1)  MD=2                                    IDØØØ71Ø
          CALL IDBVIP(MD,NCP,NDP,XD,YD,ZD,1,XI(IXI),YI(IYI),                IDØØØ72Ø
     1                ZI1(IXI,IYI),IWK,WK)                                  IDØØØ73Ø
   11   CONTINUE                                                          IDØØØ74Ø
   12 CONTINUE                                                          IDØØØ75Ø
   15 CALL IDSFFT(1,NCP,NDP,XD,YD,ZD,NXI,NYI,XI,YI,ZI2,IWK,WK)          IDØØØ76Ø
      DO 17  IYI=1,NYI                                                  IDØØØ77Ø
        DO 16  IXI=1,NXI                                                IDØØØ78Ø
          DZI1(IXI,IYI)=ABS(ZI1(IXI,IYI)-ZI(IXI,IYI))                  IDØØØ79Ø
          DZI2(IXI,IYI)=ABS(ZI2(IXI,IYI)-ZI(IXI,IYI))                  IDØØØ8ØØ
```

```
   16   CONTINUE                                                    ID000810
   17 CONTINUE                                                      ID000820
C PRINTING OF INPUT DATA                                            ID000830
   20 WRITE (LUN,2020)  NDP                                         ID000840
      DO 23  IDP=1,NDP                                              ID000850
        IF(MOD(IDP,5).EQ.1)    WRITE (LUN,2021)                     ID000860
        WRITE (LUN,2022)  IDP,XD(IDP),YD(IDP),ZD(IDP)               ID000870
   23 CONTINUE                                                      ID000880
C PRINTING OF OUTPUT RESULTS                                        ID000890
   30 WRITE (LUN,2030)                                              ID000900
      WRITE (LUN,2031)  YI                                          ID000910
      DO 33  IXI=1,NXI                                              ID000920
        WRITE (LUN,2032)  XI(IXI),(ZI1(IXI,IYI),IYI=1,NYI)          ID000930
   33 CONTINUE                                                      ID000940
   40 WRITE (LUN,2040)                                              ID000950
      WRITE (LUN,2031)  YI                                          ID000960
      DO 43  IXI=1,NXI                                              ID000970
        WRITE (LUN,2032)  XI(IXI),(DZI1(IXI,IYI),IYI=1,NYI)         ID000980
   43 CONTINUE                                                      ID000990
   50 WRITE (LUN,2050)                                              ID001000
      WRITE (LUN,2031)  YI                                          ID001010
      DO 53  IXI=1,NXI                                              ID001020
        WRITE (LUN,2032)  XI(IXI),(ZI2(IXI,IYI),IYI=1,NYI)          ID001030
   53 CONTINUE                                                      ID001040
   60 WRITE (LUN,2060)                                              ID001050
      WRITE (LUN,2031)  YI                                          ID001060
      DO 63  IXI=1,NXI                                              ID001070
        WRITE (LUN,2032)  XI(IXI),(DZI2(IXI,IYI),IYI=1,NYI)         ID001080
   63 CONTINUE                                                      ID001090
      STOP                                                          ID001100
C FORMAT STATEMENTS                                                 ID001110
 2020 FORMAT(1H1,6HTTIDBS/////3X,10HINPUT DATA,8X,5HNDP =,I3///     ID001120
     1   30H      I      XD      YD      ZD /)                      ID001130
 2021 FORMAT(1X)                                                    ID001140
 2022 FORMAT(5X,I2,2X,3F7.2)                                        ID001150
 2030 FORMAT(1H1,6HTTIDBS/////3X,17HIDBVIP SUBROUTINE///            ID001160
     1   26X,10HZI1(XI,YI))                                         ID001170
 2031 FORMAT(7X,2HXI,4X,3HYI=/12X,5F7.2/)                           ID001180
 2032 FORMAT(1X/1X,F9.2,2X,5F7.2)                                   ID001190
 2040 FORMAT(1X/////3X,10HDIFFERENCE///                             ID001200
     1   25X,11HDZI1(XI,YI))                                        ID001210
 2050 FORMAT(1H1,6HTTIDBS/////3X,17HIDSFFT SUBROUTINE///            ID001220
     1   26X,10HZI2(XI,YI))                                         ID001230
 2060 FORMAT(1X/////3X,10HDIFFERENCE///                             ID001240
     1   25X,11HDZI2(XI,YI))                                        ID001250
      END                                                           ID001260


      SUBROUTINE  IDBVIP(MD,NCP,NDP,XD,YD,ZD,NIP,XI,YI,ZI,          ID001340
     1                   IWK,WK)                                    ID001350
C THIS SUBROUTINE PERFORMS BIVARIATE INTERPOLATION WHEN THE PRO-    ID001360
C JECTIONS OF THE DATA POINTS IN THE X-Y PLANE ARE IRREGULARLY      ID001370
C DISTRIBUTED IN THE PLANE.                                         ID001380
C THE INPUT PARAMETERS ARE                                          ID001390
C     MD  = MODE OF COMPUTATION (MUST BE 1, 2, OR 3),               ID001400
C         = 1 FOR NEW NCP AND/OR NEW XD-YD,                         ID001410
C         = 2 FOR OLD NCP, OLD XD-YD, NEW XI-YI,                    ID001420
C         = 3 FOR OLD NCP, OLD XD-YD, OLD XI-YI,                    ID001430
C     NCP = NUMBER OF ADDITIONAL DATA POINTS USED FOR ESTI-         ID001440
C           MATING PARTIAL DERIVATIVES AT EACH DATA POINT           ID001450
C           (MUST BE 2 OR GREATER, BUT SMALLER THAN NDP),           ID001460
C     NDP = NUMBER OF DATA POINTS (MUST BE 4 OR GREATER),           ID001470
C     XD  = ARRAY OF DIMENSION NDP CONTAINING THE X                 ID001480
C           COORDINATES OF THE DATA POINTS,                         ID001490
C     YD  = ARRAY OF DIMENSION NDP CONTAINING THE Y                 ID001500
C           COORDINATES OF THE DATA POINTS,                         ID001510
C     ZD  = ARRAY OF DIMENSION NDP CONTAINING THE Z                 ID001520
C           COORDINATES OF THE DATA POINTS,                         ID001530
C     NIP = NUMBER OF OUTPUT POINTS AT WHICH INTERPOLATION          ID001540
C           IS TO BE PERFORMED (MUST BE 1 OR GREATER),              ID001550
C     XI  = ARRAY OF DIMENSION NIP CONTAINING THE X                 ID001560
C           COORDINATES OF THE OUTPUT POINTS,                       ID001570
C     YI  = ARRAY OF DIMENSION NIP CONTAINING THE Y                 ID001580
C           COORDINATES OF THE OUTPUT POINTS.                       ID001590
C THE OUTPUT PARAMETER IS                                           ID001600
```

```
C     ZI  = ARRAY OF DIMENSION NIP WHERE INTERPOLATED Z                 ID00161Ø
C           VALUES ARE TO BE STORED.                                    ID00162Ø
C THE OTHER PARAMETERS ARE                                              ID00163Ø
C     IWK = INTEGER ARRAY OF DIMENSION                                  ID00164Ø
C              MAXØ(31,27+NCP)*NDP+NIP                                  ID00165Ø
C           USED INTERNALLY AS A WORK AREA,                             ID00166Ø
C     WK  = ARRAY OF DIMENSION 8*NDP USED INTERNALLY AS A               ID00167Ø
C           WORK AREA.                                                  ID00168Ø
C THE VERY FIRST CALL TO THIS SUBROUTINE AND THE CALL WITH A NEW        ID00169Ø
C NCP VALUE, A NEW NDP VALUE, AND/OR NEW CONTENTS OF THE XD AND         ID00170Ø
C YD ARRAYS MUST BE MADE WITH MD=1.  THE CALL WITH MD=2 MUST BE         ID00171Ø
C PRECEDED BY ANOTHER CALL WITH THE SAME NCP AND NDP VALUES AND         ID00172Ø
C WITH THE SAME CONTENTS OF THE XD AND YD ARRAYS.  THE CALL WITH        ID00173Ø
C MD=3 MUST BE PRECEDED BY ANOTHER CALL WITH THE SAME NCP, NDP,         ID00174Ø
C AND NIP VALUES AND WITH THE SAME CONTENTS OF THE XD, YD, XI,          ID00175Ø
C AND YI ARRAYS.  BETWEEN THE CALL WITH MD=2 OR MD=3 AND ITS            ID00176Ø
C PRECEDING CALL, THE IWK AND WK ARRAYS MUST NOT BE DISTURBED.          ID00177Ø
C USE OF A VALUE BETWEEN 3 AND 5 (INCLUSIVE) FOR NCP IS RECOM-          ID00178Ø
C MENDED UNLESS THERE ARE EVIDENCES THAT DICTATE OTHERWISE.             ID00179Ø
C THE LUN CONSTANT IN THE DATA INITIALIZATION STATEMENT IS THE          ID00180Ø
C LOGICAL UNIT NUMBER OF THE STANDARD OUTPUT UNIT AND IS,               ID00181Ø
C THEREFORE, SYSTEM DEPENDENT.                                          ID00182Ø
C THIS SUBROUTINE CALLS THE IDCLDP, IDLCTN, IDPDRV, IDPTIP, AND         ID00183Ø
C IDTANG SUBROUTINES.                                                   ID00184Ø
C DECLARATION STATEMENTS                                                ID00185Ø
      DIMENSION   XD(1ØØ),YD(1ØØ),ZD(1ØØ),XI(1ØØØ),YI(1ØØØ),           ID00186Ø
     1            ZI(1ØØØ),IWK(41ØØ),WK(8ØØ)                          ID00187Ø
      COMMON/IDLC/NIT                                                   ID00188Ø
      COMMON/IDPI/ITPV                                                  ID00189Ø
      DATA  LUN/6/                                                      ID00190Ø
C SETTING OF SOME INPUT PARAMETERS TO LOCAL VARIABLES.                  ID00191Ø
C (FOR MD=1,2,3)                                                        ID00192Ø
   1Ø MDØ=MD                                                            ID00193Ø
      NCPØ=NCP                                                          ID00194Ø
      NDPØ=NDP                                                          ID00195Ø
      NIPØ=NIP                                                          ID00196Ø
C ERROR CHECK.  (FOR MD=1,2,3)                                          ID00197Ø
   2Ø IF(MDØ.LT.1.OR.MDØ.GT.3)           GO TO 9Ø                     ID00198Ø
      IF(NCPØ.LT.2.OR.NCPØ.GE.NDPØ)      GO TO 9Ø                     ID00199Ø
      IF(NDPØ.LT.4)                      GO TO 9Ø                     ID00200Ø
      IF(NIPØ.LT.1)                      GO TO 9Ø                     ID00201Ø
      IF(MDØ.GE.2)        GO TO 21                                      ID00202Ø
      IWK(1)=NCPØ                                                       ID00203Ø
      IWK(2)=NDPØ                                                       ID00204Ø
      GO TO 22                                                          ID00205Ø
   21 NCPPV=IWK(1)                                                      ID00206Ø
      NDPPV=IWK(2)                                                      ID00207Ø
      IF(NCPØ.NE.NCPPV)   GO TO 9Ø                                      ID00208Ø
      IF(NDPØ.NE.NDPPV)   GO TO 9Ø                                      ID00209Ø
   22 IF(MDØ.GE.3)        GO TO 23                                      ID00210Ø
      IWK(3)=NIP                                                        ID00211Ø
      GO TO 3Ø                                                          ID00212Ø
   23 NIPPV=IWK(3)                                                      ID00213Ø
      IF(NIPØ.NE.NIPPV)   GO TO 9Ø                                      ID00214Ø
C ALLOCATION OF STORAGE AREAS IN THE IWK ARRAY.  (FOR MD=1,2,3)         ID00215Ø
   3Ø JWIPT=16                                                          ID00216Ø
      JWIWL=6*NDPØ+1                                                    ID00217Ø
      JWIWK=JWIWL                                                       ID00218Ø
      JWIPL=24*NDPØ+1                                                   ID00219Ø
      JWIWP=3Ø*NDPØ+1                                                   ID00220Ø
      JWIPC=27*NDPØ+1                                                   ID00221Ø
      JWITØ=MAXØ(31,27+NCPØ)*NDPØ                                       ID00222Ø
C TRIANGULATES THE X-Y PLANE.  (FOR MD=1)                               ID00223Ø
   4Ø IF(MDØ.GT.1)   GO TO 5Ø                                           ID00224Ø
      CALL IDTANG(NDPØ,XD,YD,NT,IWK(JWIPT),NL,IWK(JWIPL),               ID00225Ø
     1            IWK(JWIWL),IWK(JWIWP),WK)                            ID00226Ø
      IWK(5)=NT                                                         ID00227Ø
      IWK(6)=NL                                                         ID00228Ø
      IF(NT.EQ.Ø)    RETURN                                             ID00229Ø
C DETERMINES NCP POINTS CLOSEST TO EACH DATA POINT.  (FOR MD=1)         ID00230Ø
   5Ø IF(MDØ.GT.1)   GO TO 6Ø                                           ID00231Ø
      CALL IDCLDP(NDPØ,XD,YD,NCPØ,IWK(JWIPC))                           ID00232Ø
      IF(IWK(JWIPC).EQ.Ø)      RETURN                                   ID00233Ø
C LOCATES ALL POINTS AT WHICH INTERPOLATION IS TO BE PERFORMED.          ID00234Ø
C (FOR MD=1,2)                                                          ID00235Ø
   6Ø IF(MDØ.EQ.3)   GO TO 7Ø                                           ID00236Ø
      NIT=Ø                                                             ID00237Ø
```

```
      JWIT=JWITØ                                                        IDØØ238Ø
      DO 61  IIP=1,NIPØ                                                 IDØØ239Ø
        JWIT=JWIT+1                                                     IDØØ24ØØ
        CALL IDLCTN(NDPØ,XD,YD,NT,IWK(IWIPT),NL,IWK(JWIPL),             IDØØ241Ø
     1            XI(IIP),YI(IIP),IWK(JWIT),IWK(JWIWK),WK)            IDØØ242Ø
   61 CONTINUE                                                          IDØØ243Ø
C ESTIMATES PARTIAL DERIVATIVES AT ALL DATA POINTS.                     IDØØ244Ø
C (FOR MD=1,2,3)                                                        IDØØ245Ø
   7Ø CALL IDPDRV(NDPØ,XD,YD,ZD,NCPØ,IWK(JWIPC),WK)                     IDØØ246Ø
C INTERPOLATES THE ZI VALUES.  (FOR MD=1,2,3)                           IDØØ247Ø
   8Ø ITPV=Ø                                                            IDØØ248Ø
      JWIT=JWITØ                                                        IDØØ249Ø
      DO 81  IIP=1,NIPØ                                                 IDØØ25ØØ
        JWIT=JWIT+1                                                     IDØØ251Ø
        CALL IDPTIP(XD,YD,ZD,NT,IWK(JWIPT),NL,IWK(JWIPL),WK,            IDØØ252Ø
     1              IWK(JWIT),XI(IIP),YI(IIP),ZI(IIP))                 IDØØ253Ø
   81 CONTINUE                                                          IDØØ254Ø
      RETURN                                                            IDØØ255Ø
C ERROR EXIT                                                            IDØØ256Ø
   9Ø WRITE (LUN,2Ø9Ø) MDØ,NCPØ,NDPØ,NIPØ                               IDØØ257Ø
      RETURN                                                            IDØØ258Ø
C FORMAT STATEMENT FOR ERROR MESSAGE                                    IDØØ259Ø
 2Ø9Ø FORMAT(1X/41H ***   IMPROPER INPUT PARAMETER VALUE(S)./           IDØØ26ØØ
     1   7H   MD =,I4,1ØX,5HNCP =,I6,1ØX,5HNDP =,I6,                    IDØØ261Ø
     2   1ØX,5HNIP =,I6/                                                IDØØ262Ø
     3   35H ERROR DETECTED IN ROUTINE   IDBVIP/)                       IDØØ263Ø
      END                                                               IDØØ264Ø


      SUBROUTINE  IDCLDP(NDP,XD,YD,NCP,IPC)                             IDØØ272Ø
C THIS SUBROUTINE SELECTS SEVERAL DATA POINTS THAT ARE CLOSEST          IDØØ273Ø
C TO EACH OF THE DATA POINT.                                            IDØØ274Ø
C THE INPUT PARAMETERS ARE                                              IDØØ275Ø
C     NDP = NUMBER OF DATA POINTS,                                      IDØØ276Ø
C     XD,YD = ARRAYS OF DIMENSION NDP CONTAINING THE X AND Y            IDØØ277Ø
C           COORDINATES OF THE DATA POINTS,                             IDØØ278Ø
C     NCP = NUMBER OF DATA POINTS CLOSEST TO EACH DATA                  IDØØ279Ø
C           POINTS.                                                     IDØØ28ØØ
C THE OUTPUT PARAMETER IS                                               IDØØ281Ø
C     IPC = INTEGER ARRAY OF DIMENSION NCP*NDP, WHERE THE               IDØØ282Ø
C           POINT NUMBERS OF NCP DATA POINTS CLOSEST TO                 IDØØ283Ø
C           EACH OF THE NDP DATA POINTS ARE TO BE STORED.               IDØØ284Ø
C THIS SUBROUTINE ARBITRARILY SETS A RESTRICTION THAT NCP MUST          IDØØ285Ø
C NOT EXCEED 25.                                                        IDØØ286Ø
C THE LUN CONSTANT IN THE DATA INITIALIZATION STATEMENT IS THE          IDØØ287Ø
C LOGICAL UNIT NUMBER OF THE STANDARD OUTPUT UNIT AND IS,               IDØØ288Ø
C THEREFORE, SYSTEM DEPENDENT.                                          IDØØ289Ø
C DECLARATION STATEMENTS                                                IDØØ29ØØ
      DIMENSION   XD(1ØØ),YD(1ØØ),IPC(4ØØ)                             IDØØ291Ø
      DIMENSION   DSQØ(25),IPCØ(25)                                     IDØØ292Ø
      DATA  NCPMX/25/, LUN/6/                                           IDØØ293Ø
C STATEMENT FUNCTION                                                    IDØØ294Ø
      DSQF(U1,V1,U2,V2)=(U2-U1)**2+(V2-V1)**2                           IDØØ295Ø
C PRELIMINARY PROCESSING                                                IDØØ296Ø
   1Ø NDPØ=NDP                                                          IDØØ297Ø
      NCPØ=NCP                                                          IDØØ298Ø
      IF(NDPØ.LT.2)  GO TO 9Ø                                           IDØØ299Ø
      IF(NCPØ.LT.1.OR.NCPØ.GT.NCPMX.OR.NCPØ.GE.NDPØ)    GO TO 9Ø        IDØØ3ØØØ
C CALCULATION                                                           IDØØ3Ø1Ø
   2Ø DO 59  IP1=1,NDPØ                                                 IDØØ3Ø2Ø
C - SELECTS NCP POINTS.                                                 IDØØ3Ø3Ø
        X1=XD(IP1)                                                      IDØØ3Ø4Ø
        Y1=YD(IP1)                                                      IDØØ3Ø5Ø
        J1=Ø                                                            IDØØ3Ø6Ø
        DSQMX=Ø.Ø                                                       IDØØ3Ø7Ø
        DO 22  IP2=1,NDPØ                                               IDØØ3Ø8Ø
          IF(IP2.EQ.IP1)  GO TO 22                                      IDØØ3Ø9Ø
          DSQI=DSQF(X1,Y1,XD(IP2),YD(IP2))                              IDØØ31ØØ
          J1=J1+1                                                       IDØØ311Ø
          DSQØ(J1)=DSQI                                                 IDØØ312Ø
          IPCØ(J1)=IP2                                                  IDØØ313Ø
          IF(DSQI.LE.DSQMX)    GO TO 21                                 IDØØ314Ø
          DSQMX=DSQI                                                    IDØØ315Ø
          JMX=J1                                                        IDØØ316Ø
   21     IF(J1.GE.NCPØ)  GO TO 23                                      IDØØ317Ø
   22   CONTINUE                                                        IDØØ318Ø
```

```
   23   IP2MN=IP2+1                                                    IDØØ319Ø
        IF(IP2MN.GT.NDPØ)       GO TO 3Ø                               IDØØ32ØØ
        DO 25  IP2=IP2MN,NDPØ                                          IDØØ321Ø
          IF(IP2.EQ.IP1)  GO TO 25                                     IDØØ322Ø
          DSQI=DSQF(X1,Y1,XD(IP2),YD(IP2))                             IDØØ323Ø
          IF(DSQI.GE.DSQMX)     GO TO 25                               IDØØ324Ø
          DSQØ(JMX)=DSQI                                               IDØØ325Ø
          IPCØ(JMX)=IP2                                                IDØØ326Ø
          DSQMX=Ø.Ø                                                    IDØØ327Ø
          DO 24  J1=1,NCPØ                                             IDØØ328Ø
            IF(DSQØ(J1).LE.DSQMX)   GO TO 24                           IDØØ329Ø
            DSQMX=DSQØ(J1)                                             IDØØ33ØØ
            JMX=J1                                                     IDØØ331Ø
   24     CONTINUE                                                     IDØØ332Ø
   25   CONTINUE                                                       IDØØ333Ø
C - CHECKS IF ALL THE NCP+1 POINTS ARE COLLINEAR.                      IDØØ334Ø
   3Ø   IP2=IPCØ(1)                                                    IDØØ335Ø
        DX12=XD(IP2)-X1                                                IDØØ336Ø
        DY12=YD(IP2)-Y1                                                IDØØ337Ø
        DO 31  J3=2,NCPØ                                               IDØØ338Ø
          IP3=IPCØ(J3)                                                 IDØØ339Ø
          DX13=XD(IP3)-X1                                              IDØØ34ØØ
          DY13=YD(IP3)-Y1                                              IDØØ341Ø
          IF((DY13*DX12-DX13*DY12).NE.Ø.Ø)     GO TO 5Ø                IDØØ342Ø
   31   CONTINUE                                                       IDØØ343Ø
C - SEARCHES FOR THE CLOSEST NONCOLLINEAR POINT.                       IDØØ344Ø
   4Ø   NCLPT=Ø                                                       IDØØ345Ø
        DO 43  IP3=1,NDPØ                                              IDØØ346Ø
          IF(IP3.EQ.IP1)        GO TO 43                               IDØØ347Ø
          DO 41  J4=1,NCPØ                                             IDØØ348Ø
            IF(IP3.EQ.IPCØ(J4))     GO TO 43                           IDØØ349Ø
   41     CONTINUE                                                     IDØØ35ØØ
          DX13=XD(IP3)-X1                                              IDØØ351Ø
          DY13=YD(IP3)-Y1                                              IDØØ352Ø
          IF((DY13*DX12-DX13*DY12).EQ.Ø.Ø)     GO TO 43                IDØØ353Ø
          DSQI=DSQF(X1,Y1,XD(IP3),YD(IP3))                             IDØØ354Ø
          IF(NCLPT.EQ.Ø)        GO TO 42                               IDØØ355Ø
          IF(DSQI.GE.DSQMN)     GO TO 43                               IDØØ356Ø
   42     NCLPT=1                                                      IDØØ357Ø
          DSQMN=DSQI                                                   IDØØ358Ø
          IP3MN=IP3                                                    IDØØ359Ø
   43   CONTINUE                                                       IDØØ36ØØ
        IF(NCLPT.EQ.Ø)     GO TO 91                                    IDØØ361Ø
        DSQMX=DSQMN                                                    IDØØ362Ø
        IPCØ(JMX)=IP3MN                                                IDØØ363Ø
C - REPLACES THE LOCAL ARRAY FOR THE OUTPUT ARRAY.                     IDØØ364Ø
   5Ø   J1=(IP1-1)*NCPØ                                                IDØØ365Ø
        DO 51  J2=1,NCPØ                                               IDØØ366Ø
          J1=J1+1                                                      IDØØ367Ø
          IPC(J1)=IPCØ(J2)                                             IDØØ368Ø
   51   CONTINUE                                                       IDØØ369Ø
   59 CONTINUE                                                        IDØØ37ØØ
      RETURN                                                           IDØØ371Ø
C ERROR EXIT                                                           IDØØ372Ø
   9Ø WRITE (LUN,2Ø9Ø)                                                 IDØØ373Ø
      GO TO 92                                                         IDØØ374Ø
   91 WRITE (LUN,2Ø91)                                                 IDØØ375Ø
   92 WRITE (LUN,2Ø92)  NDPØ,NCPØ                                      IDØØ376Ø
      IPC(1)=Ø                                                         IDØØ377Ø
      RETURN                                                           IDØØ378Ø
C FORMAT STATEMENTS FOR ERROR MESSAGES                                 IDØØ379Ø
 2Ø9Ø FORMAT(1X/41H ***   IMPROPER INPUT PARAMETER VALUE(S).)          IDØØ38ØØ
 2Ø91 FORMAT(1X/33H ***   ALL COLLINEAR DATA POINTS.)                  IDØØ381Ø
 2Ø92 FORMAT(8H   NDP =,I5,5X,5HNCP =,I5/                             IDØØ382Ø
     1   35H ERROR DETECTED IN ROUTINE   IDCLDP/)                      IDØØ383Ø
      END                                                              IDØØ384Ø


      SUBROUTINE IDGRID(XD, YD, NT, IPT, NL, IPL, NXI, NYI, XI, YI,    IDG   1Ø
     *  NGP, IGP)                                                      IDG   2Ø
C THIS SUBROUTINE ORGANIZES GRID POINTS FOR SURFACE FITTING BY         IDG   3Ø
C SORTING THEM IN ASCENDING ORDER OF TRIANGLE NUMBERS AND OF THE       IDG   4Ø
C BORDER LINE SEGMENT NUMBER.                                          IDG   5Ø
C THE INPUT PARAMETERS ARE                                             IDG   6Ø
C     XD,YD = ARRAYS OF DIMENSION NDP CONTAINING THE X AND Y           IDG   7Ø
C             COORDINATES OF THE DATA POINTS, WHERE NDP IS THE         IDG   8Ø
```

```
C           NUMBER OF THE DATA POINTS,                                    IDG  9Ø
C     NT  = NUMBER OF TRIANGLES,                                          IDG  1ØØ
C     IPT = INTEGER ARRAY OF DIMENSION 3*NT CONTAINING THE                IDG  11Ø
C           POINT NUMBERS OF THE VERTEXES OF THE TRIANGLES,               IDG  12Ø
C     NL  = NUMBER OF BORDER LINE SEGMENTS,                               IDG  13Ø
C     IPL = INTEGER ARRAY OF DIMENSION 3*NL CONTAINING THE                IDG  14Ø
C           POINT NUMBERS OF THE END POINTS OF THE BORDER                 IDG  15Ø
C           LINE SEGMENTS AND THEIR RESPECTIVE TRIANGLE                   IDG  16Ø
C           NUMBERS,                                                      IDG  17Ø
C     NXI = NUMBER OF GRID POINTS IN THE X COORDINATE,                    IDG  18Ø
C     NYI = NUMBER OF GRID POINTS IN THE Y COORDINATE,                    IDG  19Ø
C     XI,YI = ARRAYS OF DIMENSION NXI AND NYI CONTAINING                  IDG  2ØØ
C           THE X AND Y COORDINATES OF THE GRID POINTS,                   IDG  21Ø
C           RESPECTIVELY.                                                 IDG  22Ø
C THE OUTPUT PARAMETERS ARE                                               IDG  23Ø
C     NGP = INTEGER ARRAY OF DIMENSION 2*(NT+2*NL) WHERE THE              IDG  24Ø
C           NUMBER OF GRID POINTS THAT BELONG TO EACH OF THE              IDG  25Ø
C           TRIANGLES OR OF THE BORDER LINE SEGMENTS ARE TO               IDG  26Ø
C           BE STORED,                                                    IDG  27Ø
C     IGP = INTEGER ARRAY OF DIMENSION NXI*NYI WHERE THE                  IDG  28Ø
C           GRID POINT NUMBERS ARE TO BE STORED IN ASCENDING              IDG  29Ø
C           ORDER OF THE TRIANGLE NUMBER AND THE BORDER LINE              IDG  3ØØ
C           SEGMENT NUMBER.                                               IDG  31Ø
C DECLARATION STATEMENTS                                                  IDG  32Ø
      DIMENSION XD(1ØØ), YD(1ØØ), IPT(585), IPL(3ØØ), XI(1Ø1),            IDG  33Ø
     *  YI(1Ø1), NGP(8ØØ), IGP(1Ø2Ø1)                                     IDG  34Ø
C STATEMENT FUNCTIONS                                                     IDG  35Ø
      SIDE(U1,V1,U2,V2,U3,V3) = (U1-U3)*(V2-V3) - (V1-V3)*(U2-U3)         IDG  36Ø
      SPDT(U1,V1,U2,V2,U3,V3) = (U1-U2)*(U3-U2) + (V1-V2)*(V3-V2)         IDG  37Ø
C PRELIMINARY PROCESSING                                                  IDG  38Ø
      NTØ = NT                                                            IDG  39Ø
      NLØ = NL                                                            IDG  4ØØ
      NXIØ = NXI                                                          IDG  41Ø
      NYIØ = NYI                                                          IDG  42Ø
      NXINYI = NXIØ*NYIØ                                                  IDG  43Ø
      XIMN = AMIN1(XI(1),XI(NXIØ))                                        IDG  44Ø
      XIMX = AMAX1(XI(1),XI(NXIØ))                                        IDG  45Ø
      YIMN = AMIN1(YI(1),YI(NYIØ))                                        IDG  46Ø
      YIMX = AMAX1(YI(1),YI(NYIØ))                                        IDG  47Ø
C DETERMINES GRID POINTS INSIDE THE DATA AREA.                            IDG  48Ø
      JNGPØ = Ø                                                           IDG  49Ø
      JNGP1 = 2*(NTØ+2*NLØ) + 1                                           IDG  5ØØ
      JIGPØ = Ø                                                           IDG  51Ø
      JIGP1 = NXINYI + 1                                                  IDG  52Ø
      DO 16Ø ITØ=1,NTØ                                                    IDG  53Ø
        NGPØ = Ø                                                          IDG  54Ø
        NGP1 = Ø                                                          IDG  55Ø
        ITØT3 = ITØ*3                                                     IDG  56Ø
        IP1 = IPT(ITØT3-2)                                                IDG  57Ø
        IP2 = IPT(ITØT3-1)                                                IDG  58Ø
        IP3 = IPT(ITØT3)                                                  IDG  59Ø
        X1 = XD(IP1)                                                      IDG  6ØØ
        Y1 = YD(IP1)                                                      IDG  61Ø
        X2 = XD(IP2)                                                      IDG  62Ø
        Y2 = YD(IP2)                                                      IDG  63Ø
        X3 = XD(IP3)                                                      IDG  64Ø
        Y3 = YD(IP3)                                                      IDG  65Ø
        XMN = AMIN1(X1,X2,X3)                                             IDG  66Ø
        XMX = AMAX1(X1,X2,X3)                                             IDG  67Ø
        YMN = AMIN1(Y1,Y2,Y3)                                             IDG  68Ø
        YMX = AMAX1(Y1,Y2,Y3)                                             IDG  69Ø
        INSD = Ø                                                          IDG  7ØØ
        DO 2Ø IXI=1,NXIØ                                                  IDG  71Ø
          IF (XI(IXI).GE.XMN .AND. XI(IXI).LE.XMX) GO TO 1Ø               IDG  72Ø
          IF (INSD.EQ.Ø) GO TO 2Ø                                         IDG  73Ø
          IXIMX = IXI - 1                                                 IDG  74Ø
          GO TO 3Ø                                                        IDG  75Ø
   1Ø     IF (INSD.EQ.1) GO TO 2Ø                                         IDG  76Ø
          INSD = 1                                                        IDG  77Ø
          IXIMN = IXI                                                     IDG  78Ø
   2Ø   CONTINUE                                                          IDG  79Ø
        IF (INSD.EQ.Ø) GO TO 15Ø                                          IDG  8ØØ
        IXIMX = NXIØ                                                      IDG  81Ø
   3Ø   DO 14Ø IYI=1,NYIØ                                                 IDG  82Ø
          YII = YI(IYI)                                                   IDG  83Ø
          IF (YII.LT.YMN .OR. YII.GT.YMX) GO TO 14Ø                       IDG  84Ø
          DO 13Ø IXI=IXIMN,IXIMX                                          IDG  85Ø
```

```
            XII = XI(IXI)                                                    IDG  86Ø
            L = Ø                                                            IDG  87Ø
            IF (SIDE(X1,Y1,X2,Y2,XII,YII)) 13Ø, 4Ø, 5Ø                       IDG  88Ø
 4Ø         L = 1                                                            IDG  89Ø
 5Ø         IF (SIDE(X2,Y2,X3,Y3,XII,YII)) 13Ø, 6Ø, 7Ø                       IDG  9ØØ
 6Ø         L = 1                                                            IDG  91Ø
 7Ø         IF (SIDE(X3,Y3,X1,Y1,XII,YII)) 13Ø, 8Ø, 9Ø                       IDG  92Ø
 8Ø         L = 1                                                            IDG  93Ø
 9Ø         IZI = NXIØ*(IYI-1) + IXI                                         IDG  94Ø
            IF (L.EQ.1) GO TO 1ØØ                                            IDG  95Ø
            NGPØ = NGPØ + 1                                                  IDG  96Ø
            JIGPØ = JIGPØ + 1                                                IDG  97Ø
            IGP(JIGPØ) = IZI                                                 IDG  98Ø
            GO TO 13Ø                                                        IDG  99Ø
1ØØ         IF (JIGP1.GT.NXINYI) GO TO 12Ø                                   IDG 1ØØØ
            DO 11Ø JIGP1I=JIGP1,NXINYI                                       IDG 1Ø1Ø
              IF (IZI.EQ.IGP(JIGP1I)) GO TO 13Ø                              IDG 1Ø2Ø
11Ø         CONTINUE                                                         IDG 1Ø3Ø
12Ø         NGP1 = NGP1 + 1                                                  IDG 1Ø4Ø
            JIGP1 = JIGP1 - 1                                                IDG 1Ø5Ø
            IGP(JIGP1) = IZI                                                 IDG 1Ø6Ø
13Ø       CONTINUE                                                           IDG 1Ø7Ø
14Ø     CONTINUE                                                             IDG 1Ø8Ø
15Ø     JNGPØ = JNGPØ + 1                                                    IDG 1Ø9Ø
        NGP(JNGPØ) = NGPØ                                                    IDG 11ØØ
        JNGP1 = JNGP1 - 1                                                    IDG 111Ø
        NGP(JNGP1) = NGP1                                                    IDG 112Ø
16Ø CONTINUE                                                                 IDG 113Ø
C DETERMINES GRID POINTS OUTSIDE THE DATA AREA.                              IDG 114Ø
C - IN SEMI-INFINITE RECTANGULAR AREA.                                       IDG 115Ø
      DO 45Ø ILØ=1,NLØ                                                       IDG 116Ø
        NGPØ = Ø                                                             IDG 117Ø
        NGP1 = Ø                                                             IDG 118Ø
        ILØT3 = ILØ*3                                                        IDG 119Ø
        IP1 = IPL(ILØT3-2)                                                   IDG 12ØØ
        IP2 = IPL(ILØT3-1)                                                   IDG 121Ø
        X1 = XD(IP1)                                                         IDG 122Ø
        Y1 = YD(IP1)                                                         IDG 123Ø
        X2 = XD(IP2)                                                         IDG 124Ø
        Y2 = YD(IP2)                                                         IDG 125Ø
        XMN = XIMN                                                           IDG 126Ø
        XMX = XIMX                                                           IDG 127Ø
        YMN = YIMN                                                           IDG 128Ø
        YMX = YIMX                                                           IDG 129Ø
        IF (Y2.GE.Y1) XMN = AMIN1(X1,X2)                                     IDG 13ØØ
        IF (Y2.LE.Y1) XMX = AMAX1(X1,X2)                                     IDG 131Ø
        IF (X2.LE.X1) YMN = AMIN1(Y1,Y2)                                     IDG 132Ø
        IF (X2.GE.X1) YMX = AMAX1(Y1,Y2)                                     IDG 133Ø
        INSD = Ø                                                             IDG 134Ø
        DO 18Ø IXI=1,NXIØ                                                    IDG 135Ø
          IF (XI(IXI).GE.XMN .AND. XI(IXI).LE.XMX) GO TO 17Ø                 IDG 136Ø
          IF (INSD.EQ.Ø) GO TO 18Ø                                           IDG 137Ø
          IXIMX = IXI - 1                                                    IDG 138Ø
          GO TO 19Ø                                                          IDG 139Ø
17Ø       IF (INSD.EQ.1) GO TO 18Ø                                           IDG 14ØØ
          INSD = 1                                                           IDG 141Ø
          IXIMN = IXI                                                        IDG 142Ø
18Ø     CONTINUE                                                             IDG 143Ø
        IF (INSD.EQ.Ø) GO TO 31Ø                                             IDG 144Ø
        IXIMX = NXIØ                                                         IDG 145Ø
19Ø     DO 3ØØ IYI=1,NYIØ                                                    IDG 146Ø
          YII = YI(IYI)                                                      IDG 147Ø
          IF (YII.LT.YMN .OR. YII.GT.YMX) GO TO 3ØØ                          IDG 148Ø
          DO 29Ø IXI=IXIMN,IXIMX                                             IDG 149Ø
            XII = XI(IXI)                                                    IDG 15ØØ
            L = Ø                                                            IDG 151Ø
            IF (SIDE(X1,Y1,X2,Y2,XII,YII)) 21Ø, 2ØØ, 29Ø                     IDG 152Ø
2ØØ         L = 1                                                            IDG 153Ø
21Ø         IF (SPDT(X2,Y2,X1,Y1,XII,YII)) 29Ø, 22Ø, 23Ø                     IDG 154Ø
22Ø         L = 1                                                            IDG 155Ø
23Ø         IF (SPDT(X1,Y1,X2,Y2,XII,YII)) 29Ø, 24Ø, 25Ø                     IDG 156Ø
24Ø         L = 1                                                            IDG 157Ø
25Ø         IZI = NXIØ*(IYI-1) + IXI                                         IDG 158Ø
            IF (L.EQ.1) GO TO 26Ø                                            IDG 159Ø
            NGPØ = NGPØ + 1                                                  IDG 16ØØ
            JIGPØ = JIGPØ + 1                                                IDG 161Ø
            IGP(JIGPØ) = IZI                                                 IDG 162Ø
```

```
            GO TO 29Ø                                                    IDG 163Ø
   26Ø      IF (JIGP1.GT.NXINYI) GO TO 28Ø                               IDG 164Ø
            DO 27Ø JIGP1I=JIGP1,NXINYI                                   IDG 165Ø
              IF (IZI.EQ.IGP(JIGP1I)) GO TO 29Ø                          IDG 166Ø
   27Ø      CONTINUE                                                     IDG 167Ø
   28Ø      NGP1 = NGP1 + 1                                              IDG 168Ø
            JIGP1 = JIGP1 - 1                                            IDG 169Ø
            IGP(JIGP1) = IZI                                             IDG 17ØØ
   29Ø    CONTINUE                                                       IDG 171Ø
   3ØØ  CONTINUE                                                         IDG 172Ø
   31Ø  JNGPØ = JNGPØ + 1                                                IDG 173Ø
        NGP(JNGPØ) = NGPØ                                                IDG 174Ø
        JNGP1 = JNGP1 - 1                                                IDG 175Ø
        NGP(JNGP1) = NGP1                                                IDG 176Ø
C - IN SEMI-INFINITE TRIANGULAR AREA.                                    IDG 177Ø
        NGPØ = Ø                                                         IDG 178Ø
        NGP1 = Ø                                                         IDG 179Ø
        ILP1 = MOD(ILØ,NLØ) + 1                                          IDG 18ØØ
        ILP1T3 = ILP1*3                                                  IDG 181Ø
        IP3 = IPL(ILP1T3-1)                                              IDG 182Ø
        X3 = XD(IP3)                                                     IDG 183Ø
        Y3 = YD(IP3)                                                     IDG 184Ø
        XMN = XIMN                                                       IDG 185Ø
        XMX = XIMX                                                       IDG 186Ø
        YMN = YIMN                                                       IDG 187Ø
        YMX = YIMX                                                       IDG 188Ø
        IF (Y3.GE.Y2 .AND. Y2.GE.Y1) XMN = X2                            IDG 189Ø
        IF (Y3.LE.Y2 .AND. Y2.LE.Y1) XMX = X2                            IDG 19ØØ
        IF (X3.LE.X2 .AND. X2.LE.X1) YMN = Y2                            IDG 191Ø
        IF (X3.GE.X2 .AND. X2.GE.X1) YMX = Y2                            IDG 192Ø
        INSD = Ø                                                         IDG 193Ø
        DO 33Ø IXI=1,NXIØ                                                IDG 194Ø
          IF (XI(IXI).GE.XMN .AND. XI(IXI).LE.XMX) GO TO 32Ø             IDG 195Ø
          IF (INSD.EQ.Ø) GO TO 33Ø                                       IDG 196Ø
          IXIMX = IXI - 1                                                IDG 197Ø
          GO TO 34Ø                                                      IDG 198Ø
   32Ø    IF (INSD.EQ.1) GO TO 33Ø                                       IDG 199Ø
          INSD = 1                                                       IDG 2ØØØ
          IXIMN = IXI                                                    IDG 2Ø1Ø
   33Ø  CONTINUE                                                         IDG 2Ø2Ø
        IF (INSD.EQ.Ø) GO TO 44Ø                                         IDG 2Ø3Ø
        IXIMX = NXIØ                                                     IDG 2Ø4Ø
   34Ø  DO 43Ø IYI=1,NYIØ                                                IDG 2Ø5Ø
          YII = YI(IYI)                                                  IDG 2Ø6Ø
          IF (YII.LT.YMN .OR. YII.GT.YMX) GO TO 43Ø                      IDG 2Ø7Ø
          DO 42Ø IXI=IXIMN,IXIMX                                         IDG 2Ø8Ø
            XII = XI(IXI)                                                IDG 2Ø9Ø
            L = Ø                                                        IDG 21ØØ
            IF (SPDT(X1,Y1,X2,Y2,XII,YII)) 36Ø, 35Ø, 42Ø                 IDG 211Ø
   35Ø      L = 1                                                        IDG 212Ø
   36Ø      IF (SPDT(X3,Y3,X2,Y2,XII,YII)) 38Ø, 37Ø, 42Ø                 IDG 213Ø
   37Ø      L = 1                                                        IDG 214Ø
   38Ø      IZI = NXIØ*(IYI-1) + IXI                                     IDG 215Ø
            IF (L.EQ.1) GO TO 39Ø                                        IDG 216Ø
            NGPØ = NGPØ + 1                                              IDG 217Ø
            JIGPØ = JIGPØ + 1                                            IDG 218Ø
            IGP(JIGPØ) = IZI                                             IDG 219Ø
            GO TO 42Ø                                                    IDG 22ØØ
   39Ø      IF (JIGP1.GT.NXINYI) GO TO 41Ø                               IDG 221Ø
            DO 4ØØ JIGP1I=JIGP1,NXINYI                                   IDG 222Ø
              IF (IZI.EQ.IGP(JIGP1I)) GO TO 42Ø                          IDG 223Ø
   4ØØ      CONTINUE                                                     IDG 224Ø
   41Ø      NGP1 = NGP1 + 1                                              IDG 225Ø
            JIGP1 = JIGP1 - 1                                            IDG 226Ø
            IGP(JIGP1) = IZI                                             IDG 227Ø
   42Ø    CONTINUE                                                       IDG 228Ø
   43Ø  CONTINUE                                                         IDG 229Ø
   44Ø  JNGPØ = JNGPØ + 1                                                IDG 23ØØ
        NGP(JNGPØ) = NGPØ                                                IDG 231Ø
        JNGP1 = JNGP1 - 1                                                IDG 232Ø
        NGP(JNGP1) = NGP1                                                IDG 233Ø
   45Ø CONTINUE                                                          IDG 234Ø
       RETURN                                                            IDG 235Ø
       END                                                               IDG 236Ø
```

```
      SUBROUTINE IDLCTN(NDP, XD, YD, NT, IPT, NL, IPL, XII, YII, ITI,   IDL  1Ø
     *  IWK, WK)                                                        IDL  2Ø
C THIS SUBROUTINE LOCATES A POINT, I.E., DETERMINES TO WHAT TRI-        IDL  3Ø
C ANGLE A GIVEN POINT (XII,YII) BELONGS.  WHEN THE GIVEN POINT          IDL  4Ø
C DOES NOT LIE INSIDE THE DATA AREA, THIS SUBROUTINE DETERMINES         IDL  5Ø
C THE BORDER LINE SEGMENT WHEN THE POINT LIES IN AN OUTSIDE             IDL  6Ø
C RECTANGULAR AREA, AND TWO BORDER LINE SEGMENTS WHEN THE POINT         IDL  7Ø
C LIES IN AN OUTSIDE TRIANGULAR AREA.                                   IDL  8Ø
C THE INPUT PARAMETERS ARE                                              IDL  9Ø
C     NDP = NUMBER OF DATA POINTS,                                      IDL 1ØØ
C     XD,YD = ARRAYS OF DIMENSION NDP CONTAINING THE X AND Y            IDL 11Ø
C           COORDINATES OF THE DATA POINTS,                             IDL 12Ø
C     NT  = NUMBER OF TRIANGLES,                                        IDL 13Ø
C     IPT = INTEGER ARRAY OF DIMENSION 3*NT CONTAINING THE              IDL 14Ø
C           POINT NUMBERS OF THE VERTEXES OF THE TRIANGLES,             IDL 15Ø
C     NL  = NUMBER OF BORDER LINE SEGMENTS,                             IDL 16Ø
C     IPL = INTEGER ARRAY OF DIMENSION 3*NL CONTAINING THE              IDL 17Ø
C           POINT NUMBERS OF THE END POINTS OF THE BORDER               IDL 18Ø
C           LINE SEGMENTS AND THEIR RESPECTIVE TRIANGLE                 IDL 19Ø
C           NUMBERS,                                                    IDL 2ØØ
C     XII,YII = X AND Y COORDINATES OF THE POINT TO BE                  IDL 21Ø
C           LOCATED.                                                    IDL 22Ø
C THE OUTPUT PARAMETER IS                                               IDL 23Ø
C     ITI = TRIANGLE NUMBER, WHEN THE POINT IS INSIDE THE               IDL 24Ø
C           DATA AREA, OR                                               IDL 25Ø
C           TWO BORDER LINE SEGMENT NUMBERS, IL1 AND IL2,               IDL 26Ø
C           CODED TO IL1*(NT+NL)+IL2, WHEN THE POINT IS                 IDL 27Ø
C           OUTSIDE THE DATA AREA.                                      IDL 28Ø
C THE OTHER PARAMETERS ARE                                              IDL 29Ø
C     IWK = INTEGER ARRAY OF DIMENSION 18*NDP USED INTER-               IDL 3ØØ
C           NALLY AS A WORK AREA,                                       IDL 31Ø
C     WK  = ARRAY OF DIMENSION 8*NDP USED INTERNALLY AS A               IDL 32Ø
C           WORK AREA.                                                  IDL 33Ø
C DECLARATION STATEMENTS                                                IDL 34Ø
      DIMENSION XD(1ØØ), YD(1ØØ), IPT(585), IPL(3ØØ), IWK(18ØØ),       IDL 35Ø
     *  WK(8ØØ)                                                         IDL 36Ø
      DIMENSION NTSC(9), IDSC(9)                                        IDL 37Ø
      COMMON /IDLC/ NIT                                                 IDL 38Ø
C STATEMENT FUNCTIONS                                                   IDL 39Ø
      SIDE(U1,V1,U2,V2,U3,V3) = (U1-U3)*(V2-V3) - (V1-V3)*(U2-U3)       IDL 4ØØ
      SPDT(U1,V1,U2,V2,U3,V3) = (U1-U2)*(U3-U2) + (V1-V2)*(V3-V2)       IDL 41Ø
C PRELIMINARY PROCESSING                                                IDL 42Ø
      NDPØ = NDP                                                        IDL 43Ø
      NTØ = NT                                                          IDL 44Ø
      NLØ = NL                                                          IDL 45Ø
      NTL = NTØ + NLØ                                                   IDL 46Ø
      XØ = XII                                                          IDL 47Ø
      YØ = YII                                                          IDL 48Ø
C PROCESSING FOR A NEW SET OF DATA POINTS                               IDL 49Ø
      IF (NIT.NE.Ø) GO TO 8Ø                                            IDL 5ØØ
      NIT = 1                                                           IDL 51Ø
C - DIVIDES THE X-Y PLANE INTO NINE RECTANGULAR SECTIONS.               IDL 52Ø
      XMN = XD(1)                                                       IDL 53Ø
      XMX = XMN                                                         IDL 54Ø
      YMN = YD(1)                                                       IDL 55Ø
      YMX = YMN                                                         IDL 56Ø
      DO 1Ø IDP=2,NDPØ                                                  IDL 57Ø
        XI = XD(IDP)                                                    IDL 58Ø
        YI = YD(IDP)                                                    IDL 59Ø
        XMN = AMIN1(XI,XMN)                                             IDL 6ØØ
        XMX = AMAX1(XI,XMX)                                             IDL 61Ø
        YMN = AMIN1(YI,YMN)                                             IDL 62Ø
        YMX = AMAX1(YI,YMX)                                             IDL 63Ø
   1Ø CONTINUE                                                          IDL 64Ø
      XS1 = (XMN+XMN+XMX)/3.Ø                                           IDL 65Ø
      XS2 = (XMN+XMX+XMX)/3.Ø                                           IDL 66Ø
      YS1 = (YMN+YMN+YMX)/3.Ø                                           IDL 67Ø
      YS2 = (YMN+YMX+YMX)/3.Ø                                           IDL 68Ø
C - DETERMINES AND STORES IN THE IWK ARRAY TRIANGLE NUMBERS OF          IDL 69Ø
C - THE TRIANGLES ASSOCIATED WITH EACH OF THE NINE SECTIONS.            IDL 7ØØ
      DO 2Ø ISC=1,9                                                     IDL 71Ø
        NTSC(ISC) = Ø                                                   IDL 72Ø
        IDSC(ISC) = Ø                                                   IDL 73Ø
   2Ø CONTINUE                                                          IDL 74Ø
      ITØT3 = Ø                                                         IDL 75Ø
      JWK = Ø                                                           IDL 76Ø
```

```
      DO 7Ø ITØ=1,NTØ                                                    IDL  77Ø
        ITØT3 = ITØT3 + 3                                                IDL  78Ø
        I1 = IPT(ITØT3-2)                                                IDL  79Ø
        I2 = IPT(ITØT3-1)                                                IDL  8ØØ
        I3 = IPT(ITØT3)                                                  IDL  81Ø
        XMN = AMIN1(XD(I1),XD(I2),XD(I3))                                IDL  82Ø
        XMX = AMAX1(XD(I1),XD(I2),XD(I3))                                IDL  83Ø
        YMN = AMIN1(YD(I1),YD(I2),YD(I3))                                IDL  84Ø
        YMX = AMAX1(YD(I1),YD(I2),YD(I3))                                IDL  85Ø
        IF (YMN.GT.YS1) GO TO 3Ø                                         IDL  86Ø
        IF (XMN.LE.XS1) IDSC(1) = 1                                      IDL  87Ø
        IF (XMX.GE.XS1 .AND. XMN.LE.XS2) IDSC(2) = 1                     IDL  88Ø
        IF (XMX.GE.XS2) IDSC(3) = 1                                      IDL  89Ø
   3Ø   IF (YMX.LT.YS1 .OR. YMN.GT.YS2) GO TO 4Ø                         IDL  9ØØ
        IF (XMN.LE.XS1) IDSC(4) = 1                                      IDL  91Ø
        IF (XMX.GE.XS1 .AND. XMN.LE.XS2) IDSC(5) = 1                     IDL  92Ø
        IF (XMX.GE.XS2) IDSC(6) = 1                                      IDL  93Ø
   4Ø   IF (YMX.LT.YS2) GO TO 5Ø                                         IDL  94Ø
        IF (XMN.LE.XS1) IDSC(7) = 1                                      IDL  95Ø
        IF (XMX.GE.XS1 .AND. XMN.LE.XS2) IDSC(8) = 1                     IDL  96Ø
        IF (XMX.GE.XS2) IDSC(9) = 1                                      IDL  97Ø
   5Ø   DO 6Ø ISC=1,9                                                    IDL  98Ø
          IF (IDSC(ISC).EQ.Ø) GO TO 6Ø                                   IDL  99Ø
          JIWK = 9*NTSC(ISC) + ISC                                       IDL 1ØØØ
          IWK(JIWK) = ITØ                                                IDL 1Ø1Ø
          NTSC(ISC) = NTSC(ISC) + 1                                      IDL 1Ø2Ø
          IDSC(ISC) = Ø                                                  IDL 1Ø3Ø
   6Ø   CONTINUE                                                         IDL 1Ø4Ø
C - STORES IN THE WK ARRAY THE MINIMUM AND MAXIMUM OF THE X AND          IDL 1Ø5Ø
C - Y COORDINATE VALUES FOR EACH OF THE TRIANGLE.                        IDL 1Ø6Ø
        JWK = JWK + 4                                                    IDL 1Ø7Ø
        WK(JWK-3) = XMN                                                  IDL 1Ø8Ø
        WK(JWK-2) = XMX                                                  IDL 1Ø9Ø
        WK(JWK-1) = YMN                                                  IDL 11ØØ
        WK(JWK) = YMX                                                    IDL 111Ø
   7Ø CONTINUE                                                           IDL 112Ø
      GO TO 11Ø                                                          IDL 113Ø
C CHECKS IF IN THE SAME TRIANGLE AS PREVIOUS.                            IDL 114Ø
   8Ø ITØ = ITIPV                                                        IDL 115Ø
      IF (ITØ.GT.NTØ) GO TO 9Ø                                           IDL 116Ø
      ITØT3 = ITØ*3                                                      IDL 117Ø
      IP1 = IPT(ITØT3-2)                                                 IDL 118Ø
      X1 = XD(IP1)                                                       IDL 119Ø
      Y1 = YD(IP1)                                                       IDL 12ØØ
      IP2 = IPT(ITØT3-1)                                                 IDL 121Ø
      X2 = XD(IP2)                                                       IDL 122Ø
      Y2 = YD(IP2)                                                       IDL 123Ø
      IF (SIDE(X1,Y1,X2,Y2,XØ,YØ).LT.Ø.Ø) GO TO 11Ø                      IDL 124Ø
      IP3 = IPT(ITØT3)                                                   IDL 125Ø
      X3 = XD(IP3)                                                       IDL 126Ø
      Y3 = YD(IP3)                                                       IDL 127Ø
      IF (SIDE(X2,Y2,X3,Y3,XØ,YØ).LT.Ø.Ø) GO TO 11Ø                      IDL 128Ø
      IF (SIDE(X3,Y3,X1,Y1,XØ,YØ).LT.Ø.Ø) GO TO 11Ø                      IDL 129Ø
      GO TO 17Ø                                                          IDL 13ØØ
C CHECKS IF ON THE SAME BORDER LINE SEGMENT.                             IDL 131Ø
   9Ø IL1 = ITØ/NTL                                                      IDL 132Ø
      IL2 = ITØ - IL1*NTL                                                IDL 133Ø
      IL1T3 = IL1*3                                                      IDL 134Ø
      IP1 = IPL(IL1T3-2)                                                 IDL 135Ø
      X1 = XD(IP1)                                                       IDL 136Ø
      Y1 = YD(IP1)                                                       IDL 137Ø
      IP2 = IPL(IL1T3-1)                                                 IDL 138Ø
      X2 = XD(IP2)                                                       IDL 139Ø
      Y2 = YD(IP2)                                                       IDL 14ØØ
      IF (IL2.NE.IL1) GO TO 1ØØ                                          IDL 141Ø
      IF (SPDT(X1,Y1,X2,Y2,XØ,YØ).LT.Ø.Ø) GO TO 11Ø                      IDL 142Ø
      IF (SPDT(X2,Y2,X1,Y1,XØ,YØ).LT.Ø.Ø) GO TO 11Ø                      IDL 143Ø
      IF (SIDE(X1,Y1,X2,Y2,XØ,YØ).GT.Ø.Ø) GO TO 11Ø                      IDL 144Ø
      GO TO 17Ø                                                          IDL 145Ø
C CHECKS IF BETWEEN THE SAME TWO BORDER LINE SEGMENTS.                   IDL 146Ø
  1ØØ IF (SPDT(X1,Y1,X2,Y2,XØ,YØ).GT.Ø.Ø) GO TO 11Ø                      IDL 147Ø
      IP3 = IPL(3*IL2-1)                                                 IDL 148Ø
      X3 = XD(IP3)                                                       IDL 149Ø
      Y3 = YD(IP3)                                                       IDL 15ØØ
      IF (SPDT(X3,Y3,X2,Y2,XØ,YØ).LE.Ø.Ø) GO TO 17Ø                      IDL 151Ø
C LOCATES INSIDE THE DATA AREA.                                          IDL 152Ø
```

```
C - DETERMINES THE SECTION IN WHICH THE POINT IN QUESTION LIES.          IDL 153Ø
  11Ø ISC = 1                                                            IDL 154Ø
      IF (XØ.GE.XS1) ISC = ISC + 1                                       IDL 155Ø
      IF (XØ.GE.XS2) ISC = ISC + 1                                       IDL 156Ø
      IF (YØ.GE.YS1) ISC = ISC + 3                                       IDL 157Ø
      IF (YØ.GE.YS2) ISC = ISC + 3                                       IDL 158Ø
C - SEARCHES THROUGH THE TRIANGLES ASSOCIATED WITH THE SECTION.          IDL 159Ø
      NTSCI = NTSC(ISC)                                                  IDL 16ØØ
      IF (NTSCI.LE.Ø) GO TO 13Ø                                          IDL 161Ø
      JIWK = -9 + ISC                                                    IDL 162Ø
      DO 12Ø ITSC=1,NTSCI                                                IDL 163Ø
        JIWK = JIWK + 9                                                  IDL 164Ø
        ITØ = IWK(JIWK)                                                  IDL 165Ø
        JWK = ITØ*4                                                      IDL 166Ø
        IF (XØ.LT.WK(JWK-3)) GO TO 12Ø                                   IDL 167Ø
        IF (XØ.GT.WK(JWK-2)) GO TO 12Ø                                   IDL 168Ø
        IF (YØ.LT.WK(JWK-1)) GO TO 12Ø                                   IDL 169Ø
        IF (YØ.GT.WK(JWK)) GO TO 12Ø                                     IDL 17ØØ
        ITØT3 = ITØ*3                                                    IDL 171Ø
        IP1 = IPT(ITØT3-2)                                               IDL 172Ø
        X1 = XD(IP1)                                                     IDL 173Ø
        Y1 = YD(IP1)                                                     IDL 174Ø
        IP2 = IPT(ITØT3-1)                                               IDL 175Ø
        X2 = XD(IP2)                                                     IDL 176Ø
        Y2 = YD(IP2)                                                     IDL 177Ø
        IF (SIDE(X1,Y1,X2,Y2,XØ,YØ).LT.Ø.Ø) GO TO 12Ø                    IDL 178Ø
        IP3 = IPT(ITØT3)                                                 IDL 179Ø
        X3 = XD(IP3)                                                     IDL 18ØØ
        Y3 = YD(IP3)                                                     IDL 181Ø
        IF (SIDE(X2,Y2,X3,Y3,XØ,YØ).LT.Ø.Ø) GO TO 12Ø                    IDL 182Ø
        IF (SIDE(X3,Y3,X1,Y1,XØ,YØ).LT.Ø.Ø) GO TO 12Ø                    IDL 183Ø
        GO TO 17Ø                                                        IDL 184Ø
  12Ø CONTINUE                                                           IDL 185Ø
C LOCATES OUTSIDE THE DATA AREA.                                         IDL 186Ø
  13Ø DO 15Ø IL1=1,NLØ                                                   IDL 187Ø
        IL1T3 = IL1*3                                                    IDL 188Ø
        IP1 = IPL(IL1T3-2)                                               IDL 189Ø
        X1 = XD(IP1)                                                     IDL 19ØØ
        Y1 = YD(IP1)                                                     IDL 191Ø
        IP2 = IPL(IL1T3-1)                                               IDL 192Ø
        X2 = XD(IP2)                                                     IDL 193Ø
        Y2 = YD(IP2)                                                     IDL 194Ø
        IF (SPDT(X2,Y2,X1,Y1,XØ,YØ).LT.Ø.Ø) GO TO 15Ø                    IDL 195Ø
        IF (SPDT(X1,Y1,X2,Y2,XØ,YØ).LT.Ø.Ø) GO TO 14Ø                    IDL 196Ø
        IF (SIDE(X1,Y1,X2,Y2,XØ,YØ).GT.Ø.Ø) GO TO 15Ø                    IDL 197Ø
        IL2 = IL1                                                        IDL 198Ø
        GO TO 16Ø                                                        IDL 199Ø
  14Ø   IL2 = MOD(IL1,NLØ) + 1                                           IDL 2ØØØ
        IP3 = IPL(3*IL2-1)                                               IDL 2Ø1Ø
        X3 = XD(IP3)                                                     IDL 2Ø2Ø
        Y3 = YD(IP3)                                                     IDL 2Ø3Ø
        IF (SPDT(X3,Y3,X2,Y2,XØ,YØ).LE.Ø.Ø) GO TO 16Ø                    IDL 2Ø4Ø
  15Ø CONTINUE                                                           IDL 2Ø5Ø
      ITØ = 1                                                            IDL 2Ø6Ø
      GO TO 17Ø                                                          IDL 2Ø7Ø
  16Ø ITØ = IL1*NTL + IL2                                                IDL 2Ø8Ø
C NORMAL EXIT                                                            IDL 2Ø9Ø
  17Ø ITI = ITØ                                                          IDL 21ØØ
      ITIPV = ITØ                                                        IDL 211Ø
      RETURN                                                             IDL 212Ø
      END                                                                IDL 213Ø


      SUBROUTINE  IDPDRV(NDP,XD,YD,ZD,NCP,IPC,PD)                        IDØØ894Ø
C THIS SUBROUTINE ESTIMATES PARTIAL DERIVATIVES OF THE FIRST AND         IDØØ895Ø
C SECOND ORDER AT THE DATA POINTS.                                       IDØØ896Ø
C THE INPUT PARAMETERS ARE                                               IDØØ897Ø
C     NDP = NUMBER OF DATA POINTS,                                       IDØØ898Ø
C     XD,YD,ZD = ARRAYS OF DIMENSION NDP CONTAINING THE X,               IDØØ899Ø
C           Y, AND Z COORDINATES OF THE DATA POINTS,                     IDØØ9ØØØ
C     NCP = NUMBER OF ADDITIONAL DATA POINTS USED FOR ESTI-              IDØØ9Ø1Ø
C           MATING PARTIAL DERIVATIVES AT EACH DATA POINT,               IDØØ9Ø2Ø
C     IPC = INTEGER ARRAY OF DIMENSION NCP*NDP CONTAINING                IDØØ9Ø3Ø
C           THE POINT NUMBERS OF NCP DATA POINTS CLOSEST TO              IDØØ9Ø4Ø
C           EACH OF THE NDP DATA POINTS.                                 IDØØ9Ø5Ø
C THE OUTPUT PARAMETER IS                                                IDØØ9Ø6Ø
```

```
C     PD  = ARRAY OF DIMENSION 5*NDP, WHERE THE ESTIMATED          ID00907Ø
C           ZX, ZY, ZXX, ZXY, AND ZYY VALUES AT THE DATA           ID00908Ø
C           POINTS ARE TO BE STORED.                               ID00909Ø
C DECLARATION STATEMENTS                                           ID00910Ø
      DIMENSION   XD(1ØØ),YD(1ØØ),ZD(1ØØ),IPC(4ØØ),PD(5ØØ)       ID00911Ø
      REAL        NMX,NMY,NMZ,NMXX,NMXY,NMYX,NMYY                ID00912Ø
C PRELIMINARY PROCESSING                                           ID00913Ø
   1Ø NDPØ=NDP                                                     ID00914Ø
      NCPØ=NCP                                                     ID00915Ø
      NCPM1=NCPØ-1                                                 ID00916Ø
C ESTIMATION OF ZX AND ZY                                          ID00917Ø
   2Ø DO 24  IPØ=1,NDPØ                                            ID00918Ø
        XØ=XD(IPØ)                                                 ID00919Ø
        YØ=YD(IPØ)                                                 ID00920Ø
        ZØ=ZD(IPØ)                                                 ID00921Ø
        NMX=Ø.Ø                                                    ID00922Ø
        NMY=Ø.Ø                                                    ID00923Ø
        NMZ=Ø.Ø                                                    ID00924Ø
        JIPCØ=NCPØ*(IPØ-1)                                         ID00925Ø
        DO 23  IC1=1,NCPM1                                         ID00926Ø
          JIPC=JIPCØ+IC1                                           ID00927Ø
          IPI=IPC(JIPC)                                            ID00928Ø
          DX1=XD(IPI)-XØ                                           ID00929Ø
          DY1=YD(IPI)-YØ                                           ID00930Ø
          DZ1=ZD(IPI)-ZØ                                           ID00931Ø
          IC2MN=IC1+1                                              ID00932Ø
          DO 22  IC2=IC2MN,NCPØ                                    ID00933Ø
            JIPC=JIPCØ+IC2                                         ID00934Ø
            IPI=IPC(JIPC)                                          ID00935Ø
            DX2=XD(IPI)-XØ                                         ID00936Ø
            DY2=YD(IPI)-YØ                                         ID00937Ø
            DNMZ=DX1*DY2-DY1*DX2                                   ID00938Ø
            IF(DNMZ.EQ.Ø.Ø)    GO TO 22                           ID00939Ø
            DZ2=ZD(IPI)-ZØ                                         ID00940Ø
            DNMX=DY1*DZ2-DZ1*DY2                                   ID00941Ø
            DNMY=DZ1*DX2-DX1*DZ2                                   ID00942Ø
            IF(DNMZ.GE.Ø.Ø)    GO TO 21                           ID00943Ø
            DNMX=-DNMX                                             ID00944Ø
            DNMY=-DNMY                                             ID00945Ø
            DNMZ=-DNMZ                                             ID00946Ø
   21       NMX=NMX+DNMX                                           ID00947Ø
            NMY=NMY+DNMY                                           ID00948Ø
            NMZ=NMZ+DNMZ                                           ID00949Ø
   22     CONTINUE                                                 ID00950Ø
   23   CONTINUE                                                   ID00951Ø
        JPDØ=5*IPØ                                                 ID00952Ø
        PD(JPDØ-4)=-NMX/NMZ                                        ID00953Ø
        PD(JPDØ-3)=-NMY/NMZ                                        ID00954Ø
   24 CONTINUE                                                     ID00955Ø
C ESTIMATION OF ZXX, ZXY, AND ZYY                                  ID00956Ø
   3Ø DO 34  IPØ=1,NDPØ                                            ID00957Ø
        JPDØ=JPDØ+5                                                ID00958Ø
        XØ=XD(IPØ)                                                 ID00959Ø
        JPDØ=5*IPØ                                                 ID00960Ø
        YØ=YD(IPØ)                                                 ID00961Ø
        ZXØ=PD(JPDØ-4)                                             ID00962Ø
        ZYØ=PD(JPDØ-3)                                             ID00963Ø
        NMXX=Ø.Ø                                                   ID00964Ø
        NMXY=Ø.Ø                                                   ID00965Ø
        NMYX=Ø.Ø                                                   ID00966Ø
        NMYY=Ø.Ø                                                   ID00967Ø
        NMZ =Ø.Ø                                                   ID00968Ø
        JIPCØ=NCPØ*(IPØ-1)                                         ID00969Ø
        DO 33  IC1=1,NCPM1                                         ID00970Ø
          JIPC=JIPCØ+IC1                                           ID00971Ø
          IPI=IPC(JIPC)                                            ID00972Ø
          DX1=XD(IPI)-XØ                                           ID00973Ø
          DY1=YD(IPI)-YØ                                           ID00974Ø
          JPD=5*IPI                                                ID00975Ø
          DZX1=PD(JPD-4)-ZXØ                                       ID00976Ø
          DZY1=PD(JPD-3)-ZYØ                                       ID00977Ø
          IC2MN=IC1+1                                              ID00978Ø
          DO 32  IC2=IC2MN,NCPØ                                    ID00979Ø
            JIPC=JIPCØ+IC2                                         ID00980Ø
            IPI=IPC(JIPC)                                          ID00981Ø
            DX2=XD(IPI)-XØ                                         ID00982Ø
```

```
            DY2=YD(IPI)-YØ                                              IDØØ983Ø
            DNMZ =DX1*DY2 -DY1*DX2                                      IDØØ984Ø
            IF(DNMZ.EQ.Ø.Ø)     GO TO 32                                IDØØ985Ø
            JPD=5*IPI                                                   IDØØ986Ø
            DZX2=PD(JPD-4)-ZXØ                                          IDØØ987Ø
            DZY2=PD(JPD-3)-ZYØ                                          IDØØ988Ø
            DNMXX=DY1*DZX2-DZX1*DY2                                     IDØØ989Ø
            DNMXY=DZX1*DX2-DX1*DZX2                                     IDØØ99ØØ
            DNMYX=DY1*DZY2-DZY1*DY2                                     IDØØ991Ø
            DNMYY=DZY1*DX2-DX1*DZY2                                     IDØØ992Ø
            IF(DNMZ.GE.Ø.Ø)     GO TO 31                                IDØØ993Ø
            DNMXX=-DNMXX                                                IDØØ994Ø
            DNMXY=-DNMXY                                                IDØØ995Ø
            DNMYX=-DNMYX                                                IDØØ996Ø
            DNMYY=-DNMYY                                                IDØØ997Ø
            DNMZ =-DNMZ                                                 IDØØ998Ø
   31       NMXX=NMXX+DNMXX                                             IDØØ999Ø
            NMXY=NMXY+DNMXY                                             IDØ1ØØØØ
            NMYX=NMYX+DNMYX                                             IDØ1ØØ1Ø
            NMYY=NMYY+DNMYY                                             IDØ1ØØ2Ø
            NMZ =NMZ +DNMZ                                              IDØ1ØØ3Ø
   32     CONTINUE                                                      IDØ1ØØ4Ø
   33   CONTINUE                                                        IDØ1ØØ5Ø
        PD(JPDØ-2)=-NMXX/NMZ                                            IDØ1ØØ6Ø
        PD(JPDØ-1)=-(NMXY+NMYX)/(2.Ø*NMZ)                               IDØ1ØØ7Ø
        PD(JPDØ)  =-NMYY/NMZ                                            IDØ1ØØ8Ø
   34 CONTINUE                                                          IDØ1ØØ9Ø
      RETURN                                                            IDØ1Ø1Ø
      END                                                               IDØ1Ø11Ø


      SUBROUTINE  IDPTIP(XD,YD,ZD,NT,IPT,NL,IPL,PDD,ITI,XII,YII,        IDØ1Ø19Ø
     1                   ZII)                                           IDØ1Ø2ØØ
C THIS SUBROUTINE PERFORMS PUNCTUAL INTERPOLATION OR EXTRAPOLA-         IDØ1Ø21Ø
C TION, I.E., DETERMINES THE Z VALUE AT A POINT.                        IDØ1Ø22Ø
C THE INPUT PARAMETERS ARE                                              IDØ1Ø23Ø
C     XD,YD,ZD = ARRAYS OF DIMENSION NDP CONTAINING THE X,              IDØ1Ø24Ø
C           Y, AND Z COORDINATES OF THE DATA POINTS, WHERE              IDØ1Ø25Ø
C           NDP IS THE NUMBER OF THE DATA POINTS,                       IDØ1Ø26Ø
C     NT  = NUMBER OF TRIANGLES,                                        IDØ1Ø27Ø
C     IPT = INTEGER ARRAY OF DIMENSION 3*NT CONTAINING THE              IDØ1Ø28Ø
C           POINT NUMBERS OF THE VERTEXES OF THE TRIANGLES,             IDØ1Ø29Ø
C     NL  = NUMBER OF BORDER LINE SEGMENTS,                             IDØ1Ø3ØØ
C     IPL = INTEGER ARRAY OF DIMENSION 3*NL CONTAINING THE              IDØ1Ø31Ø
C           POINT NUMBERS OF THE END POINTS OF THE BORDER               IDØ1Ø32Ø
C           LINE SEGMENTS AND THEIR RESPECTIVE TRIANGLE                 IDØ1Ø33Ø
C           NUMBERS,                                                    IDØ1Ø34Ø
C     PDD = ARRAY OF DIMENSION 5*NDP CONTAINING THE PARTIAL             IDØ1Ø35Ø
C           DERIVATIVES AT THE DATA POINTS,                             IDØ1Ø36Ø
C     ITI = TRIANGLE NUMBER OF THE TRIANGLE IN WHICH LIES               IDØ1Ø37Ø
C           THE POINT FOR WHICH INTERPOLATION IS TO BE                  IDØ1Ø38Ø
C           PERFORMED,                                                  IDØ1Ø39Ø
C     XII,YII = X AND Y COORDINATES OF THE POINT FOR WHICH              IDØ1Ø4ØØ
C           INTERPOLATION IS TO BE PERFORMED.                           IDØ1Ø41Ø
C THE OUTPUT PARAMETER IS                                               IDØ1Ø42Ø
C     ZII = INTERPOLATED Z VALUE.                                       IDØ1Ø43Ø
C DECLARATION STATEMENTS                                                IDØ1Ø44Ø
      DIMENSION   XD(1ØØ),YD(1ØØ),ZD(1ØØ),IPT(585),IPL(3ØØ),           IDØ1Ø45Ø
     1            PDD(5ØØ)                                            IDØ1Ø46Ø
      COMMON/IDPI/ITPV                                                  IDØ1Ø47Ø
      DIMENSION   X(3),Y(3),Z(3),PD(15),                                IDØ1Ø48Ø
     1            ZU(3),ZV(3),ZUU(3),ZUV(3),ZVV(3)                    IDØ1Ø49Ø
      REAL        LU,LV                                                 IDØ1Ø5ØØ
      EQUIVALENCE (P5,P5Ø)                                              IDØ1Ø51Ø
C PRELIMINARY PROCESSING                                                IDØ1Ø52Ø
   1Ø ITØ=ITI                                                           IDØ1Ø53Ø
      NTL=NT+NL                                                         IDØ1Ø54Ø
      IF(ITØ.LE.NTL)      GO TO 2Ø                                      IDØ1Ø55Ø
      IL1=ITØ/NTL                                                       IDØ1Ø56Ø
      IL2=ITØ-IL1*NTL                                                   IDØ1Ø57Ø
      IF(IL1.EQ.IL2)      GO TO 4Ø                                      IDØ1Ø58Ø
      GO TO 6Ø                                                          IDØ1Ø59Ø
C CALCULATION OF ZII BY INTERPOLATION.                                  IDØ1Ø6ØØ
C CHECKS IF THE NECESSARY COEFFICIENTS HAVE BEEN CALCULATED.            IDØ1Ø61Ø
   2Ø IF(ITØ.EQ.ITPV)     GO TO 3Ø                                      IDØ1Ø62Ø
C LOADS COORDINATE AND PARTIAL DERIVATIVE VALUES AT THE                 IDØ1Ø63Ø
```

```
C VERTEXES.                                                          ID01064Ø
   21 JIPT=3*(IT0-1)                                                 ID01065Ø
      JPD=Ø                                                          ID01066Ø
      DO 23  I=1,3                                                   ID01067Ø
        JIPT=JIPT+1                                                  ID01068Ø
        IDP=IPT(JIPT)                                                ID01069Ø
        X(I)=XD(IDP)                                                 ID01070Ø
        Y(I)=YD(IDP)                                                 ID01071Ø
        Z(I)=ZD(IDP)                                                 ID01072Ø
        JPDD=5*(IDP-1)                                               ID01073Ø
        DO 22  KPD=1,5                                               ID01074Ø
          JPD=JPD+1                                                  ID01075Ø
          JPDD=JPDD+1                                                ID01076Ø
          PD(JPD)=PDD(JPDD)                                          ID01077Ø
   22   CONTINUE                                                     ID01078Ø
   23 CONTINUE                                                       ID01079Ø
C DETERMINES THE COEFFICIENTS FOR THE COORDINATE SYSTEM              ID01080Ø
C TRANSFORMATION FROM THE X-Y SYSTEM TO THE U-V SYSTEM               ID01081Ø
C AND VICE VERSA.                                                    ID01082Ø
   24 XØ=X(1)                                                        ID01083Ø
      YØ=Y(1)                                                        ID01084Ø
      A=X(2)-XØ                                                      ID01085Ø
      B=X(3)-XØ                                                      ID01086Ø
      C=Y(2)-YØ                                                      ID01087Ø
      D=Y(3)-YØ                                                      ID01088Ø
      AD=A*D                                                         ID01089Ø
      BC=B*C                                                         ID01090Ø
      DLT=AD-BC                                                      ID01091Ø
      AP= D/DLT                                                      ID01092Ø
      BP=-B/DLT                                                      ID01093Ø
      CP=-C/DLT                                                      ID01094Ø
      DP= A/DLT                                                      ID01095Ø
C CONVERTS THE PARTIAL DERIVATIVES AT THE VERTEXES OF THE            ID01096Ø
C TRIANGLE FOR THE U-V COORDINATE SYSTEM.                            ID01097Ø
   25 AA=A*A                                                         ID01098Ø
      ACT2=2.Ø*A*C                                                   ID01099Ø
      CC=C*C                                                         ID01100Ø
      AB=A*B                                                         ID01101Ø
      ADBC=AD+BC                                                     ID01102Ø
      CD=C*D                                                         ID01103Ø
      BB=B*B                                                         ID01104Ø
      BDT2=2.Ø*B*D                                                   ID01105Ø
      DD=D*D                                                         ID01106Ø
      DO 26  I=1,3                                                   ID01107Ø
        IPD=5*I                                                      ID01108Ø
        ZU(I)=A*PD(JPD-4)+C*PD(JPD-3)                                ID01109Ø
        ZV(I)=B*PD(JPD-4)+D*PD(JPD-3)                                ID01110Ø
        ZUU(I)=AA*PD(JPD-2)+ACT2*PD(JPD-1)+CC*PD(JPD)                ID01111Ø
        ZUV(I)=AB*PD(JPD-2)+ADBC*PD(JPD-1)+CD*PD(JPD)                ID01112Ø
        ZVV(I)=BB*PD(JPD-2)+BDT2*PD(JPD-1)+DD*PD(JPD)                ID01113Ø
   26 CONTINUE                                                       ID01114Ø
C CALCULATES THE COEFFICIENTS OF THE POLYNOMIAL.                     ID01115Ø
   27 PØØ=Z(1)                                                       ID01116Ø
      P1Ø=ZU(1)                                                      ID01117Ø
      PØ1=ZV(1)                                                      ID01118Ø
      P2Ø=Ø.5*ZUU(1)                                                 ID01119Ø
      P11=ZUV(1)                                                     ID01120Ø
      PØ2=Ø.5*ZVV(1)                                                 ID01121Ø
      H1=Z(2)-PØØ-P1Ø-P2Ø                                            ID01122Ø
      H2=ZU(2)-P1Ø-ZUU(1)                                            ID01123Ø
      H3=ZUU(2)-ZUU(1)                                               ID01124Ø
      P3Ø= 1Ø.Ø*H1-4.Ø*H2+Ø.5*H3                                     ID01125Ø
      P4Ø=-15.Ø*H1+7.Ø*H2    -H3                                     ID01126Ø
      P5Ø=  6.Ø*H1-3.Ø*H2+Ø.5*H3                                     ID01127Ø
      H1=Z(3)-PØØ-PØ1-PØ2                                            ID01128Ø
      H2=ZV(3)-PØ1-ZVV(1)                                            ID01129Ø
      H3=ZVV(3)-ZVV(1)                                               ID01130Ø
      PØ3= 1Ø.Ø*H1-4.Ø*H2+Ø.5*H3                                     ID01131Ø
      PØ4=-15.Ø*H1+7.Ø*H2    -H3                                     ID01132Ø
      PØ5=  6.Ø*H1-3.Ø*H2+Ø.5*H3                                     ID01133Ø
      LU=SQRT(AA+CC)                                                 ID01134Ø
      LV=SQRT(BB+DD)                                                 ID01135Ø
      THXU=ATAN2(C,A)                                                ID01136Ø
      THUV=ATAN2(D,B)-THXU                                           ID01137Ø
      CSUV=COS(THUV)                                                 ID01138Ø
      P41=5.Ø*LV*CSUV/LU*P5Ø                                         ID01139Ø
```

```
      P14=5.Ø*LU*CSUV/LV*PØ5                                          IDØ114ØØ
      H1=ZV(2)-PØ1-P11-P41                                            IDØ1141Ø
      H2=ZUV(2)-P11-4.Ø*P41                                           IDØ1142Ø
      P21= 3.Ø*H1-H2                                                  IDØ1143Ø
      P31=-2.Ø*H1+H2                                                  IDØ1144Ø
      H1=ZU(3)-P1Ø-P11-P14                                            IDØ1145Ø
      H2=ZUV(3)-P11-4.Ø*P14                                           IDØ1146Ø
      P12= 3.Ø*H1-H2                                                  IDØ1147Ø
      P13=-2.Ø*H1+H2                                                  IDØ1148Ø
      THUS=ATAN2(D-C,B-A)-THXU                                        IDØ1149Ø
      THSV=THUV-THUS                                                  IDØ115ØØ
      AA= SIN(THSV)/LU                                                IDØ1151Ø
      BB=-COS(THSV)/LU                                                IDØ1152Ø
      CC= SIN(THUS)/LV                                                IDØ1153Ø
      DD= COS(THUS)/LV                                                IDØ1154Ø
      AC=AA*CC                                                        IDØ1155Ø
      AD=AA*DD                                                        IDØ1156Ø
      BC=BB*CC                                                        IDØ1157Ø
      G1=AA*AC*(3.Ø*BC+2.Ø*AD)                                        IDØ1158Ø
      G2=CC*AC*(3.Ø*AD+2.Ø*BC)                                        IDØ1159Ø
      H1=-AA*AA*AA*(5.Ø*AA*BB*P5Ø+(4.Ø*BC+AD)*P41)                    IDØ116ØØ
     1   -CC*CC*CC*(5.Ø*CC*DD*PØ5+(4.Ø*AD+BC)*P14)                    IDØ1161Ø
      H2=Ø.5*ZVV(2)-PØ2-P12                                           IDØ1162Ø
      H3=Ø.5*ZUU(3)-P2Ø-P21                                           IDØ1163Ø
      P22=(G1*H2+G2*H3-H1)/(G1+G2)                                    IDØ1164Ø
      P32=H2-P22                                                      IDØ1165Ø
      P23=H3-P22                                                      IDØ1166Ø
      ITPV=ITØ                                                        IDØ1167Ø
C CONVERTS XII AND YII TO U-V SYSTEM.                                 IDØ1168Ø
   3Ø DX=XII-XØ                                                       IDØ1169Ø
      DY=YII-YØ                                                       IDØ117ØØ
      U=AP*DX+BP*DY                                                   IDØ1171Ø
      V=CP*DX+DP*DY                                                   IDØ1172Ø
C EVALUATES THE POLYNOMIAL.                                           IDØ1173Ø
   31 PØ=PØØ+V*(PØ1+V*(PØ2+V*(PØ3+V*(PØ4+V*PØ5))))                    IDØ1174Ø
      P1=P1Ø+V*(P11+V*(P12+V*(P13+V*P14)))                            IDØ1175Ø
      P2=P2Ø+V*(P21+V*(P22+V*P23))                                    IDØ1176Ø
      P3=P3Ø+V*(P31+V*P32)                                            IDØ1177Ø
      P4=P4Ø+V*P41                                                    IDØ1178Ø
      ZII=PØ+U*(P1+U*(P2+U*(P3+U*(P4+U*P5))))                         IDØ1179Ø
      RETURN                                                          IDØ118ØØ
C CALCULATION OF ZII BY EXTRAPOLATION IN THE RECTANGLE.               IDØ1181Ø
C CHECKS IF THE NECESSARY COEFFICIENTS HAVE BEEN CALCULATED.          IDØ1182Ø
   4Ø IF(ITØ.EQ.ITPV)     GO TO 5Ø                                    IDØ1183Ø
C LOADS COORDINATE AND PARTIAL DERIVATIVE VALUES AT THE END           IDØ1184Ø
C POINTS OF THE BORDER LINE SEGMENT.                                  IDØ1185Ø
   41 JIPL=3*(IL1-1)                                                  IDØ1186Ø
      JPD=Ø                                                           IDØ1187Ø
      DO 43  I=1,2                                                    IDØ1188Ø
        JIPL=JIPL+1                                                   IDØ1189Ø
        IDP=IPL(JIPL)                                                 IDØ119ØØ
        X(I)=XD(IDP)                                                  IDØ1191Ø
        Y(I)=YD(IDP)                                                  IDØ1192Ø
        Z(I)=ZD(IDP)                                                  IDØ1193Ø
        JPDD=5*(IDP-1)                                                IDØ1194Ø
        DO 42  KPD=1,5                                                IDØ1195Ø
          JPD=JPD+1                                                   IDØ1196Ø
          JPDD=JPDD+1                                                 IDØ1197Ø
          PD(JPD)=PDD(JPDD)                                           IDØ1198Ø
   42   CONTINUE                                                      IDØ1199Ø
   43 CONTINUE                                                        IDØ12ØØØ
C DETERMINES THE COEFFICIENTS FOR THE COORDINATE SYSTEM               IDØ12Ø1Ø
C TRANSFORMATION FROm THE X-Y SYSTEM TO THE U-V SYSTEM                IDØ12Ø2Ø
C AND VICE VERSA.                                                     IDØ12Ø3Ø
   44 XØ=X(1)                                                         IDØ12Ø4Ø
      YØ=Y(1)                                                         IDØ12Ø5Ø
      A=Y(2)-Y(1)                                                     IDØ12Ø6Ø
      B=X(2)-X(1)                                                     IDØ12Ø7Ø
      C=-B                                                            IDØ12Ø8Ø
      D=A                                                             IDØ12Ø9Ø
      AD=A*D                                                          IDØ121ØØ
      BC=B*C                                                          IDØ1211Ø
      DLT=AD-BC                                                       IDØ1212Ø
      AP= D/DLT                                                       IDØ1213Ø
      BP=-B/DLT                                                       IDØ1214Ø
      CP=-BP                                                          IDØ1215Ø
```

```
      DP= AP                                                          ID01216Ø
C CONVERTS THE PARTIAL DERIVATIVES AT THE END POINTS OF THE           ID01217Ø
C BORDER LINE SEGMENT FOR THE U-V COORDINATE SYSTEM.                  ID01218Ø
   45 AA=A*A                                                          ID01219Ø
      ACT2=2.Ø*A*C                                                    ID01220Ø
      CC=C*C                                                          ID01221Ø
      AB=A*B                                                          ID01222Ø
      ADBC=AD+BC                                                      ID01223Ø
      CD=C*D                                                          ID01224Ø
      BB=B*B                                                          ID01225Ø
      BDT2=2.Ø*B*D                                                    ID01226Ø
      DD=D*D                                                          ID01227Ø
      DO 46  I=1,2                                                    ID01228Ø
        JPD=5*I                                                       ID01229Ø
        ZU(I)=A*PD(JPD-4)+C*PD(JPD-3)                                 ID01230Ø
        ZV(I)=B*PD(JPD-4)+D*PD(JPD-3)                                 ID01231Ø
        ZUU(I)=AA*PD(JPD-2)+ACT2*PD(JPD-1)+CC*PD(JPD)                 ID01232Ø
        ZUV(I)=AB*PD(JPD-2)+ADBC*PD(JPD-1)+CD*PD(JPD)                 ID01233Ø
        ZVV(I)=BB*PD(JPD-2)+BDT2*PD(JPD-1)+DD*PD(JPD)                 ID01234Ø
   46 CONTINUE                                                        ID01235Ø
C CALCULATES THE COEFFICIENTS OF THE POLYNOMIAL.                      ID01236Ø
   47 PØØ=Z(1)                                                        ID01237Ø
      P1Ø=ZU(1)                                                       ID01238Ø
      PØ1=ZV(1)                                                       ID01239Ø
      P2Ø=Ø.5*ZUU(1)                                                  ID01240Ø
      P11=ZUV(1)                                                      ID01241Ø
      PØ2=Ø.5*ZVV(1)                                                  ID01242Ø
      H1=Z(2)-PØØ-PØ1-PØ2                                             ID01243Ø
      H2=ZV(2)-PØ1-ZVV(1)                                             ID01244Ø
      H3=ZVV(2)-ZVV(1)                                                ID01245Ø
      PØ3= 1Ø.Ø*H1-4.Ø*H2+Ø.5*H3                                     ID01246Ø
      PØ4=-15.Ø*H1+7.Ø*H2    -H3                                      ID01247Ø
      PØ5=  6.Ø*H1-3.Ø*H2+Ø.5*H3                                      ID01248Ø
      H1=ZU(2)-P1Ø-P11                                                ID01249Ø
      H2=ZUV(2)-P11                                                   ID01250Ø
      P12= 3.Ø*H1-H2                                                  ID01251Ø
      P13=-2.Ø*H1+H2                                                  ID01252Ø
      P21=Ø.Ø                                                         ID01253Ø
      P23=-ZUU(2)+ZUU(1)                                              ID01254Ø
      P22=-1.5*P23                                                    ID01255Ø
      ITPV=ITØ                                                        ID01256Ø
C CONVERTS XII AND YII TO U-V SYSTEM.                                 ID01257Ø
   5Ø DX=XII-XØ                                                       ID01258Ø
      DY=YII-YØ                                                       ID01259Ø
      U=AP*DX+BP*DY                                                   ID01260Ø
      V=CP*DX+DP*DY                                                   ID01261Ø
C EVALUATES THE POLYNOMIAL.                                           ID01262Ø
   51 PØ=PØØ+V*(PØ1+V*(PØ2+V*(PØ3+V*(PØ4+V*PØ5))))                   ID01263Ø
      P1=P1Ø+V*(P11+V*(P12+V*P13))                                    ID01264Ø
      P2=P2Ø+V*(P21+V*(P22+V*P23))                                    ID01265Ø
      ZII=PØ+U*(P1+U*P2)                                              ID01266Ø
      RETURN                                                          ID01267Ø
C CALCULATION OF ZII BY EXTRAPOLATION IN THE TRIANGLE.                ID01268Ø
C CHECKS IF THE NECESSARY COEFFICIENTS HAVE BEEN CALCULATED.          ID01269Ø
   6Ø IF(ITØ.EQ.ITPV)     GO TO 7Ø                                    ID01270Ø
C LOADS COORDINATE AND PARTIAL DERIVATIVE VALUES AT THE VERTEX        ID01271Ø
C OF THE TRIANGLE.                                                    ID01272Ø
   61 JIPL=3*IL2-2                                                    ID01273Ø
      IDP=IPL(JIPL)                                                   ID01274Ø
      X(1)=XD(IDP)                                                    ID01275Ø
      Y(1)=YD(IDP)                                                    ID01276Ø
      Z(1)=ZD(IDP)                                                    ID01277Ø
      JPDD=5*(IDP-1)                                                  ID01278Ø
      DO 62  KPD=1,5                                                  ID01279Ø
        JPDD=JPDD+1                                                   ID01280Ø
        PD(KPD)=PDD(JPDD)                                             ID01281Ø
   62 CONTINUE                                                        ID01282Ø
C CALCULATES THE COEFFICIENTS OF THE POLYNOMIAL.                      ID01283Ø
   67 PØØ=Z(1)                                                        ID01284Ø
      P1Ø=PD(1)                                                       ID01285Ø
      PØ1=PD(2)                                                       ID01286Ø
      P2Ø=Ø.5*PD(3)                                                   ID01287Ø
      P11=PD(4)                                                       ID01288Ø
      PØ2=Ø.5*PD(5)                                                   ID01289Ø
      ITPV=ITØ                                                        ID01290Ø
C CONVERTS XII AND YII TO U-V SYSTEM.                                 ID01291Ø
```

```
   7Ø U=XII-X(1)                                                    IDØ1292Ø
      V=YII-Y(1)                                                    IDØ1293Ø
C EVALUATES THE POLYNOMIAL.                                         IDØ1294Ø
   71 PØ=PØØ+V*(PØ1+V*PØ2)                                          IDØ1295Ø
      P1=P1Ø+V*P11                                                  IDØ1296Ø
      ZII=PØ+U*(P1+U*P2Ø)                                           IDØ1297Ø
      RETURN                                                        IDØ1298Ø
      END                                                           IDØ1299Ø


      SUBROUTINE  IDSFFT(MD,NCP,NDP,XD,YD,ZD,NXI,NYI,XI,YI,ZI,      IDØ13Ø7Ø
     1                   IWK,WK)                                    IDØ13Ø8Ø
C THIS SUBROUTINE PERFORMS SMOOTH SURFACE FITTING WHEN THE PRO-     IDØ13Ø9Ø
C JECTIONS OF THE DATA POINTS IN THE X-Y PLANE ARE IRREGULARLY      IDØ131ØØ
C DISTRIBUTED IN THE PLANE.                                         IDØ1311Ø
C THE INPUT PARAMETERS ARE                                          IDØ1312Ø
C     MD  = MODE OF COMPUTATION (MUST BE 1, 2, OR 3),               IDØ1313Ø
C         = 1 FOR NEW NCP AND/OR NEW XD-YD,                         IDØ1314Ø
C         = 2 FOR OLD NCP, OLD XD-YD, NEW XI-YI,                    IDØ1315Ø
C         = 3 FOR OLD NCP, OLD XD-YD, OLD XI-YI,                    IDØ1316Ø
C     NCP = NUMBER OF ADDITIONAL DATA POINTS USED FOR ESTI-         IDØ1317Ø
C           MATING PARTIAL DERIVATIVES AT EACH DATA POINT           IDØ1318Ø
C           (MUST BE 2 OR GREATER, BUT SMALLER THAN NDP),           IDØ1319Ø
C     NDP = NUMBER OF DATA POINTS (MUST BE 4 OR GREATER),           IDØ132ØØ
C     XD  = ARRAY OF DIMENSION NDP CONTAINING THE X                 IDØ1321Ø
C           COORDINATES OF THE DATA POINTS,                         IDØ1322Ø
C     YD  = ARRAY OF DIMENSION NDP CONTAINING THE Y                 IDØ1323Ø
C           COORDINATES OF THE DATA POINTS,                         IDØ1324Ø
C     ZD  = ARRAY OF DIMENSION NDP CONTAINING THE Z                 IDØ1325Ø
C           COORDINATES OF THE DATA POINTS,                         IDØ1326Ø
C     NXI = NUMBER OF OUTPUT GRID POINTS IN THE X COORDINATE        IDØ1327Ø
C           (MUST BE 1 OR GREATER),                                 IDØ1328Ø
C     NYI = NUMBER OF OUTPUT GRID POINTS IN THE Y COORDINATE        IDØ1329Ø
C           (MUST BE 1 OR GREATER),                                 IDØ133ØØ
C     XI  = ARRAY OF DIMENSION NXI CONTAINING THE X                 IDØ1331Ø
C           COORDINATES OF THE OUTPUT GRID POINTS,                  IDØ1332Ø
C     YI  = ARRAY OF DIMENSION NYI CONTAINING THE Y                 IDØ1333Ø
C           COORDINATES OF THE OUTPUT GRID POINTS.                  IDØ1334Ø
C THE OUTPUT PARAMETER IS                                           IDØ1335Ø
C     ZI  = DOUBLY-DIMENSIONED ARRAY OF DIMENSION (NXI,NYI),        IDØ1336Ø
C           WHERE THE INTERPOLATED Z VALUES AT THE OUTPUT           IDØ1337Ø
C           GRID POINTS ARE TO BE STORED.                           IDØ1338Ø
C THE OTHER PARAMETERS ARE                                          IDØ1339Ø
C     IWK = INTEGER ARRAY OF DIMENSION                              IDØ134ØØ
C             MAXØ(31,27+NCP)*NDP+NXI*NYI                         IDØ1341Ø
C           USED INTERNALLY AS A WORK AREA,                         IDØ1342Ø
C     WK  = ARRAY OF DIMENSION 5*NDP USED INTERNALLY AS A           IDØ1343Ø
C           WORK AREA.                                              IDØ1344Ø
C THE VERY FIRST CALL TO THIS SUBROUTINE AND THE CALL WITH A NEW    IDØ1345Ø
C NCP VALUE, A NEW NDP VALUE, AND/OR NEW CONTENTS OF THE XD AND     IDØ1346Ø
C YD ARRAYS MUST BE MADE WITH MD=1.  THE CALL WITH MD=2 MUST BE     IDØ1347Ø
C PRECEDED BY ANOTHER CALL WITH THE SAME NCP AND NDP VALUES AND     IDØ1348Ø
C WITH THE SAME CONTENTS OF THE XD AND YD ARRAYS.  THE CALL WITH    IDØ1349Ø
C MD=3 MUST BE PRECEDED BY ANOTHER CALL WITH THE SAME NCP, NDP,     IDØ135ØØ
C NXI, AND NYI VALUES AND WITH THE SAME CONTENTS OF THE XD, YD,     IDØ1351Ø
C XI, AND YI ARRAYS.  BETWEEN THE CALL WITH MD=2 OR MD=3 AND ITS    IDØ1352Ø
C PRECEDING CALL, THE IWK AND WK ARRAYS MUST NOT BE DISTURBED.      IDØ1353Ø
C USE OF A VALUE BETWEEN 3 AND 5 (INCLUSIVE) FOR NCP IS RECOM-      IDØ1354Ø
C MENDED UNLESS THERE ARE EVIDENCES THAT DICTATE OTHERWISE.         IDØ1355Ø
C THE LUN CONSTANT IN THE DATA INITIALIZATION STATEMENT IS THE      IDØ1356Ø
C LOGICAL UNIT NUMBER OF THE STANDARD OUTPUT UNIT AND IS,           IDØ1357Ø
C THEREFORE, SYSTEM DEPENDENT.                                      IDØ1358Ø
C THIS SUBROUTINE CALLS THE IDCLDP, IDGRID, IDPDRV, IDPTIP, AND     IDØ1359Ø
C IDTANG SUBROUTINES.                                               IDØ136ØØ
C DECLARATION STATEMENTS                                            IDØ1361Ø
      DIMENSION   XD(1ØØ),YD(1ØØ),ZD(1ØØ),XI(1Ø1),YI(1Ø1),         IDØ1362Ø
     1            ZI(1Ø2Ø1),IWK(133Ø1),WK(5ØØ)                    IDØ1363Ø
      COMMON/IDPI/ITPV                                              IDØ1364Ø
      DATA  LUN/6/                                                  IDØ1365Ø
C SETTING OF SOME INPUT PARAMETERS TO LOCAL VARIABLES.              IDØ1366Ø
C (FOR MD=1,2,3)                                                    IDØ1367Ø
   1Ø MDØ=MD                                                        IDØ1368Ø
      NCPØ=NCP                                                      IDØ1369Ø
      NDPØ=NDP                                                      IDØ137ØØ
      NXIØ=NXI                                                      IDØ1371Ø
      NYIØ=NYI                                                      IDØ1372Ø
```

```
C ERROR CHECK.  (FOR MD=1,2,3)                                          ID01373Ø
   2Ø IF(MDØ.LT.1.OR.MDØ.GT.3)          GO TO 9Ø                        ID01374Ø
      IF(NCPØ.LT.2.OR.NCPØ.GE.NDPØ)     GO TO 9Ø                        ID01375Ø
      IF(NDPØ.LT.4)                     GO TO 9Ø                        ID01376Ø
      IF(NXIØ.LT.1.OR.NYIØ.LT.1)        GO TO 9Ø                        ID01377Ø
      IF(MDØ.GE.2)        GO TO 21                                        ID01378Ø
      IWK(1)=NCPØ                                                         ID01379Ø
      IWK(2)=NDPØ                                                         ID01380Ø
      GO TO 22                                                            ID01381Ø
   21 NCPPV=IWK(1)                                                        ID01382Ø
      NDPPV=IWK(2)                                                        ID01383Ø
      IF(NCPØ.NE.NCPPV)   GO TO 9Ø                                        ID01384Ø
      IF(NDPØ.NE.NDPPV)   GO TO 9Ø                                        ID01385Ø
   22 IF(MDØ.GE.3)        GO TO 23                                        ID01386Ø
      IWK(3)=NXIØ                                                         ID01387Ø
      IWK(4)=NYIØ                                                         ID01388Ø
      GO TO 3Ø                                                            ID01389Ø
   23 NXIPV=IWK(3)                                                        ID01390Ø
      NYIPV=IWK(4)                                                        ID01391Ø
      IF(NXIØ.NE.NXIPV)   GO TO 9Ø                                        ID01392Ø
      IF(NYIØ.NE.NYIPV)   GO TO 9Ø                                        ID01393Ø
C ALLOCATION OF STORAGE AREAS IN THE IWK ARRAY.  (FOR MD=1,2,3)           ID01394Ø
   3Ø JWIPT=16                                                            ID01395Ø
      JWIWL=6*NDPØ+1                                                      ID01396Ø
      JWNGPØ=JWIWL-1                                                      ID01397Ø
      JWIPL=24*NDPØ+1                                                     ID01398Ø
      JWIWP=3Ø*NDPØ+1                                                     ID01399Ø
      JWIPC=27*NDPØ+1                                                     ID01400Ø
      JWIGPØ=MAXØ(31,27+NCPØ)*NDPØ                                        ID01401Ø
C TRIANGULATES THE X-Y PLANE.  (FOR MD=1)                                 ID01402Ø
   4Ø IF(MDØ.GT.1)   GO TO 5Ø                                             ID01403Ø
      CALL IDTANG(NDPØ,XD,YD,NT,IWK(JWIPT),NL,IWK(JWIPL),                 ID01404Ø
     1            IWK(JWIWL),IWK(JWIWP),WK)                               ID01405Ø
      IWK(5)=NT                                                           ID01406Ø
      IWK(6)=NL                                                           ID01407Ø
      IF(NT.EQ.Ø)    RETURN                                               ID01408Ø
C DETERMINES NCP POINTS CLOSEST TO EACH DATA POINT.  (FOR MD=1)           ID01409Ø
   5Ø IF(MDØ.GT.1)   GO TO 6Ø                                             ID01410Ø
      CALL IDCLDP(NDPØ,XD,YD,NCPØ,IWK(JWIPC))                             ID01411Ø
      IF(IWK(JWIPC).EQ.Ø)      RETURN                                     ID01412Ø
C SORTS OUTPUT GRID POINTS IN ASCENDING ORDER OF THE TRIANGLE             ID01413Ø
C NUMBER AND THE BORDER LINE SEGMENT NUMBER.  (FOR MD=1,2)                ID01414Ø
   6Ø IF(MDØ.EQ.3)   GO TO 7Ø                                             ID01415Ø
      CALL IDGRID(XD,YD,NT,IWK(JWIPT),NL,IWK(JWIPL),NXIØ,NYIØ,            ID01416Ø
     1            XI,YI,IWK(JWNGPØ+1),IWK(JWIGPØ+1))                      ID01417Ø
C ESTIMATES PARTIAL DERIVATIVES AT ALL DATA POINTS.                       ID01418Ø
C (FOR MD=1,2,3)                                                          ID01419Ø
   7Ø CALL IDPDRV(NDPØ,XD,YD,ZD,NCPØ,IWK(JWIPC),WK)                       ID01420Ø
C INTERPOLATES THE ZI VALUES.  (FOR MD=1,2,3)                             ID01421Ø
   8Ø ITPV=Ø                                                              ID01422Ø
      JIGØMX=Ø                                                            ID01423Ø
      JIG1MN=NXIØ*NYIØ+1                                                  ID01424Ø
      NNGP=NT+2*NL                                                        ID01425Ø
      DO 89  JNGP=1,NNGP                                                  ID01426Ø
        ITI=JNGP                                                          ID01427Ø
        IF(JNGP.LE.NT)     GO TO 81                                       ID01428Ø
        IL1=(JNGP-NT+1)/2                                                 ID01429Ø
        IL2=(JNGP-NT+2)/2                                                 ID01430Ø
        IF(IL2.GT.NL)      IL2=1                                          ID01431Ø
        ITI=IL1*(NT+NL)+IL2                                               ID01432Ø
   81   JWNGP=JWNGPØ+JNGP                                                 ID01433Ø
        NGPØ=IWK(JWNGP)                                                   ID01434Ø
        IF(NGPØ.EQ.Ø)      GO TO 86                                       ID01435Ø
        JIGØMN=JIGØMX+1                                                   ID01436Ø
        JIGØMX=JIGØMX+NGPØ                                                ID01437Ø
        DO 82  JIGP=JIGØMN,JIGØMX                                         ID01438Ø
          JWIGP=JWIGPØ+JIGP                                               ID01439Ø
          IZI=IWK(JWIGP)                                                  ID01440Ø
          IYI=(IZI-1)/NXIØ+1                                              ID01441Ø
          IXI=IZI-NXIØ*(IYI-1)                                            ID01442Ø
          CALL IDPTIP(XD,YD,ZD,NT,IWK(JWIPT),NL,IWK(JWIPL),WK,            ID01443Ø
     1                ITI,XI(IXI),YI(IYI),ZI(IZI))                        ID01444Ø
   82   CONTINUE                                                          ID01445Ø
   86   JWNGP=JWNGPØ+2*NNGP+1-JNGP                                        ID01446Ø
        NGP1=IWK(JWNGP)                                                   ID01447Ø
        IF(NGP1.EQ.Ø)      GO TO 89                                       ID01448Ø
```

```
      JIG1MX=JIG1MN-1                                                    ID01449Ø
      JIG1MN=JIG1MN-NGP1                                                 ID01450Ø
      DO 87  JIGP=JIG1MN,JIG1MX                                          ID01451Ø
        JWIGP=JWIGPØ+JIGP                                                ID01452Ø
        IZI=IWK(JWIGP)                                                   ID01453Ø
        IYI=(IZI-1)/NXIØ+1                                               ID01454Ø
        IXI=IZI-NXIØ*(IYI-1)                                             ID01455Ø
        CALL IDPTIP(XD,YD,ZD,NT,IWK(JWIPT),NL,IWK(JWIPL),WK,             ID01456Ø
     1              ITI,XI(IXI),YI(IYI),ZI(IZI))                         ID01457Ø
   87   CONTINUE                                                         ID01458Ø
   89 CONTINUE                                                          ID01459Ø
      RETURN                                                             ID01460Ø
C ERROR EXIT                                                             ID01461Ø
   9Ø WRITE (LUN,2Ø9Ø) MDØ,NCPØ,NDPØ,NXIØ,NYIØ                          ID01462Ø
      RETURN                                                             ID01463Ø
C FORMAT STATEMENT FOR ERROR MESSAGE                                     ID01464Ø
 2Ø9Ø FORMAT(1X/41H ***   IMPROPER INPUT PARAMETER VALUE(S)./             ID01465Ø
     1   7H   MD =,I4,1ØX,5HNCP =,I6,1ØX,5HNDP =,I6,                     ID01466Ø
     2   1ØX,5HNXI =,I6,1ØX,5HNYI =,I6/                                  ID01467Ø
     3   35H ERROR DETECTED IN ROUTINE   IDSFFT/)                        ID01468Ø
      END                                                                ID01469Ø


      SUBROUTINE  IDTANG(NDP,XD,YD,NT,IPT,NL,IPL,IWL,IWP,WK)             ID01477Ø
C THIS SUBROUTINE PERFORMS TRIANGULATION.  IT DIVIDES THE X-Y            ID01478Ø
C PLANE INTO A NUMBER OF TRIANGLES ACCORDING TO GIVEN DATA               ID01479Ø
C POINTS IN THE PLANE, DETERMINES LINE SEGMENTS THAT FORM THE            ID01480Ø
C BORDER OF DATA AREA, AND DETERMINES THE TRIANGLE NUMBERS               ID01481Ø
C CORRESPONDING TO THE BORDER LINE SEGMENTS.                             ID01482Ø
C AT COMPLETION, POINT NUMBERS OF THE VERTEXES OF EACH TRIANGLE          ID01483Ø
C ARE LISTED COUNTER-CLOCKWISE.  POINT NUMBERS OF THE END POINTS         ID01484Ø
C OF EACH BORDER LINE SEGMENT ARE LISTED COUNTER-CLOCKWISE,              ID01485Ø
C LISTING ORDER OF THE LINE SEGMENTS BEING COUNTER-CLOCKWISE.            ID01486Ø
C THE LUN CONSTANT IN THE DATA INITIALIZATION STATEMENT IS THE           ID01487Ø
C LOGICAL UNIT NUMBER OF THE STANDARD OUTPUT UNIT AND IS,                ID01488Ø
C THEREFORE, SYSTEM DEPENDENT.                                           ID01489Ø
C THIS SUBROUTINE CALLS THE IDXCHG FUNCTION.                             ID01490Ø
C THE INPUT PARAMETERS ARE                                               ID01491Ø
C     NDP = NUMBER OF DATA POINTS,                                       ID01492Ø
C     XD  = ARRAY OF DIMENSION NDP CONTAINING THE                        ID01493Ø
C           X COORDINATES OF THE DATA POINTS,                            ID01494Ø
C     YD  = ARRAY OF DIMENSION NDP CONTAINING THE                        ID01495Ø
C           Y COORDINATES OF THE DATA POINTS.                            ID01496Ø
C THE OUTPUT PARAMETERS ARE                                              ID01497Ø
C     NT  = NUMBER OF TRIANGLES,                                         ID01498Ø
C     IPT = INTEGER ARRAY OF DIMENSION 6*NDP-15, WHERE THE               ID01499Ø
C           POINT NUMBERS OF THE VERTEXES OF THE (IT)TH                  ID01500Ø
C           TRIANGLE ARE TO BE STORED AS THE (3*IT-2)ND,                 ID01501Ø
C           (3*IT-1)ST, AND (3*IT)TH ELEMENTS,                           ID01502Ø
C           IT=1,2,...,NT,                                               ID01503Ø
C     NL  = NUMBER OF BORDER LINE SEGMENTS,                              ID01504Ø
C     IPL = INTEGER ARRAY OF DIMENSION 6*NDP, WHERE THE                  ID01505Ø
C           POINT NUMBERS OF THE END POINTS OF THE (IL)TH                ID01506Ø
C           BORDER LINE SEGMENT AND ITS RESPECTIVE TRIANGLE              ID01507Ø
C           NUMBER ARE TO BE STORED AS THE (3*IL-2)ND,                   ID01508Ø
C           (3*IL-1)ST, AND (3*IL)TH ELEMENTS,                           ID01509Ø
C           IL=1,2,..., NL.                                              ID01510Ø
C THE OTHER PARAMETERS ARE                                               ID01511Ø
C     IWL = INTEGER ARRAY OF DIMENSION 18*NDP USED                       ID01512Ø
C           INTERNALLY AS A WORK AREA,                                   ID01513Ø
C     IWP = INTEGER ARRAY OF DIMENSION NDP USED                          ID01514Ø
C           INTERNALLY AS A WORK AREA,                                   ID01515Ø
C     WK  = ARRAY OF DIMENSION NDP USED INTERNALLY AS A                  ID01516Ø
C           WORK AREA.                                                   ID01517Ø
C DECLARATION STATEMENTS                                                 ID01518Ø
      DIMENSION   XD(1ØØ),YD(1ØØ),IPT(585),IPL(6ØØ),                    ID01519Ø
     1            IWL(18ØØ),IWP(1ØØ),WK(1ØØ)                           ID01520Ø
      DIMENSION   ITF(2)                                                 ID01521Ø
      DATA  RATIO/1.ØE-6/, NREP/1ØØ/, LUN/6/                             ID01522Ø
C STATEMENT FUNCTIONS                                                    ID01523Ø
      DSQF(U1,V1,U2,V2)=(U2-U1)**2+(V2-V1)**2                            ID01524Ø
      SIDE(U1,V1,U2,V2,U3,V3)=(V3-V1)*(U2-U1)-(U3-U1)*(V2-V1)            ID01525Ø
C PRELIMINARY PROCESSING                                                 ID01526Ø
   1Ø NDPØ=NDP                                                           ID01527Ø
      NDPM1=NDPØ-1                                                       ID01528Ø
      IF(NDPØ.LT.4)       GO TO 9Ø                                       ID01529Ø
C DETERMINES THE CLOSEST PAIR OF DATA POINTS AND THEIR MIDPOINT.         ID01530Ø
   2Ø DSQMN=DSQF(XD(1),YD(1),XD(2),YD(2))                                ID01531Ø
```

```
      IPMN1=1                                                          ID01532Ø
      IPMN2=2                                                          ID01533Ø
      DO 22  IP1=1,NDPM1                                               ID01534Ø
        X1=XD(IP1)                                                     ID01535Ø
        Y1=YD(IP1)                                                     ID01536Ø
        IP1P1=IP1+1                                                    ID01537Ø
        DO 21  IP2=IP1P1,NDPØ                                          ID01538Ø
          DSQI=DSQF(X1,Y1,XD(IP2),YD(IP2))                             ID01539Ø
          IF(DSQI.EQ.Ø.Ø)       GO TO 91                               ID01540Ø
          IF(DSQI.GE.DSQMN)     GO TO 21                               ID01541Ø
          DSQMN=DSQI                                                   ID01542Ø
          IPMN1=IP1                                                    ID01543Ø
          IPMN2=IP2                                                    ID01544Ø
   21   CONTINUE                                                       ID01545Ø
   22 CONTINUE                                                        ID01546Ø
      DSQ12=DSQMN                                                      ID01547Ø
      XDMP=(XD(IPMN1)+XD(IPMN2))/2.Ø                                   ID01548Ø
      YDMP=(YD(IPMN1)+YD(IPMN2))/2.Ø                                   ID01549Ø
C SORTS THE OTHER (NDP-2) DATA POINTS IN ASCENDING ORDER OF            ID01550Ø
C DISTANCE FROM THE MIDPOINT AND STORES THE SORTED DATA POINT          ID01551Ø
C NUMBERS IN THE IWP ARRAY.                                            ID01552Ø
   3Ø JP1=2                                                            ID01553Ø
      DO 31  IP1=1,NDPØ                                                ID01554Ø
        IF(IP1.EQ.IPMN1.OR.IP1.EQ.IPMN2)      GO TO 31                 ID01555Ø
        JP1=JP1+1                                                      ID01556Ø
        IWP(JP1)=IP1                                                   ID01557Ø
        WK(JP1)=DSQF(XDMP,YDMP,XD(IP1),YD(IP1))                        ID01558Ø
   31 CONTINUE                                                         ID01559Ø
      DO 33  JP1=3,NDPM1                                               ID01560Ø
        DSQMN=WK(JP1)                                                  ID01561Ø
        JPMN=JP1                                                       ID01562Ø
        DO 32  JP2=JP1,NDPØ                                            ID01563Ø
          IF(WK(JP2).GE.DSQMN)      GO TO 32                           ID01564Ø
          DSQMN=WK(JP2)                                                ID01565Ø
          JPMN=JP2                                                     ID01566Ø
   32   CONTINUE                                                       ID01567Ø
        ITS=IWP(JP1)                                                   ID01568Ø
        IWP(JP1)=IWP(JPMN)                                             ID01569Ø
        IWP(JPMN)=ITS                                                  ID01570Ø
        WK(JPMN)=WK(JP1)                                               ID01571Ø
   33 CONTINUE                                                         ID01572Ø
C IF NECESSARY, MODIFIES THE ORDERING IN SUCH A WAY THAT THE           ID01573Ø
C FIRST THREE DATA POINTS ARE NOT COLLINEAR.                           ID01574Ø
   35 AR=DSQ12*RATIO                                                   ID01575Ø
      X1=XD(IPMN1)                                                     ID01576Ø
      Y1=YD(IPMN1)                                                     ID01577Ø
      DX21=XD(IPMN2)-X1                                                ID01578Ø
      DY21=YD(IPMN2)-Y1                                                ID01579Ø
      DO 36  JP=3,NDPØ                                                 ID01580Ø
        IP=IWP(JP)                                                     ID01581Ø
        IF(ABS((YD(IP)-Y1)*DX21-(XD(IP)-X1)*DY21).GT.AR)               ID01582Ø
     1               GO TO 37                                          ID01583Ø
   36 CONTINUE                                                         ID01584Ø
      GO TO 92                                                         ID01585Ø
   37 IF(JP.EQ.3)    GO TO 4Ø                                         ID01586Ø
      JPMX=JP                                                          ID01587Ø
      JP=JPMX+1                                                        ID01588Ø
      DO 38  JP=4,JPMX                                                 ID01589Ø
        JP=JP-1                                                        ID01590Ø
        IWP(JP)=IWP(JP-1)                                              ID01591Ø
   38 CONTINUE                                                         ID01592Ø
      IWP(3)=IP                                                        ID01593Ø
C FORMS THE FIRST TRIANGLE.  STORES POINT NUMBERS OF THE VER-          ID01594Ø
C TEXES OF THE TRIANGLE IN THE IPT ARRAY, AND STORES POINT NUM-        ID01595Ø
C BERS OF THE BORDER LINE SEGMENTS AND THE TRIANGLE NUMBER IN          ID01596Ø
C THE IPL ARRAY.                                                       ID01597Ø
   4Ø IP1=IPMN1                                                        ID01598Ø
      IP2=IPMN2                                                        ID01599Ø
      IP3=IWP(3)                                                       ID01600Ø
      IF(SIDE(XD(IP1),YD(IP1),XD(IP2),YD(IP2),XD(IP3),YD(IP3))         ID01601Ø
     1     .GE.Ø.Ø)       GO TO 41                                     ID01602Ø
      IP1=IPMN2                                                        ID01603Ø
      IP2=IPMN1                                                        ID01604Ø
   41 NTØ=1                                                            ID01605Ø
      NTT3=3                                                           ID01606Ø
      IPT(1)=IP1                                                       ID01607Ø
```

```
      IPT(2)=IP2                                                        ID01608Ø
      IPT(3)=IP3                                                        ID01609Ø
      NLØ=3                                                             ID01610Ø
      NLT3=9                                                            ID01611Ø
      IPL(1)=IP1                                                        ID01612Ø
      IPL(2)=IP2                                                        ID01613Ø
      IPL(3)=1                                                          ID01614Ø
      IPL(4)=IP2                                                        ID01615Ø
      IPL(5)=IP3                                                        ID01616Ø
      IPL(6)=1                                                          ID01617Ø
      IPL(7)=IP3                                                        ID01618Ø
      IPL(8)=IP1                                                        ID01619Ø
      IPL(9)=1                                                          ID01620Ø
C ADDS THE REMAINING (NDP-3) DATA POINTS, ONE BY ONE.                   ID01621Ø
   5Ø DO 79  JP1=4,NDPØ                                                 ID01622Ø
        IP1=IWP(JP1)                                                    ID01623Ø
        X1=XD(IP1)                                                      ID01624Ø
        Y1=YD(IP1)                                                      ID01625Ø
C - DETERMINES THE VISIBLE BORDER LINE SEGMENTS.                        ID01626Ø
        IP2=IPL(1)                                                      ID01627Ø
        JPMN=1                                                          ID01628Ø
        DXMN=XD(IP2)-X1                                                 ID01629Ø
        DYMN=YD(IP2)-Y1                                                 ID01630Ø
        DSQMN=DXMN**2+DYMN**2                                           ID01631Ø
        ARMN=DSQMN*RATIO                                                ID01632Ø
        JPMX=1                                                          ID01633Ø
        DXMX=DXMN                                                       ID01634Ø
        DYMX=DYMN                                                       ID01635Ø
        DSQMX=DSQMN                                                     ID01636Ø
        ARMX=ARMN                                                       ID01637Ø
        DO 52  JP2=2,NLØ                                                ID01638Ø
          IP2=IPL(3*JP2-2)                                              ID01639Ø
          DX=XD(IP2)-X1                                                 ID01640Ø
          DY=YD(IP2)-Y1                                                 ID01641Ø
          AR=DY*DXMN-DX*DYMN                                            ID01642Ø
          IF(AR.GT.ARMN)        GO TO 51                                ID01643Ø
          DSQI=DX**2+DY**2                                              ID01644Ø
          IF(AR.GE.(-ARMN).AND.DSQI.GE.DSQMN)       GO TO 51            ID01645Ø
          JPMN=JP2                                                      ID01646Ø
          DXMN=DX                                                       ID01647Ø
          DYMN=DY                                                       ID01648Ø
          DSQMN=DSQI                                                    ID01649Ø
          ARMN=DSQMN*RATIO                                              ID01650Ø
   51     AR=DY*DXMX-DX*DYMX                                            ID01651Ø
          IF(AR.LT.(-ARMX))     GO TO 52                                ID01652Ø
          DSQI=DX**2+DY**2                                              ID01653Ø
          IF(AR.LE.ARMX.AND.DSQI.GE.DSQMX)     GO TO 52                 ID01654Ø
          JPMX=JP2                                                      ID01655Ø
          DXMX=DX                                                       ID01656Ø
          DYMX=DY                                                       ID01657Ø
          DSQMX=DSQI                                                    ID01658Ø
          ARMX=DSQMX*RATIO                                              ID01659Ø
   52   CONTINUE                                                        ID01660Ø
        IF(JPMX.LT.JPMN)  JPMX=JPMX+NLØ                                 ID01661Ø
        NSH=JPMN-1                                                      ID01662Ø
        IF(NSH.LE.Ø)       GO TO 6Ø                                     ID01663Ø
C - SHIFTS (ROTATES) THE IPL ARRAY TO HAVE THE INVISIBLE BORDER          ID01664Ø
C - LINE SEGMENTS CONTAINED IN THE FIRST PART OF THE IPL ARRAY.         ID01665Ø
        NSHT3=NSH*3                                                     ID01666Ø
        DO 53  JP2T3=3,NSHT3,3                                          ID01667Ø
          JP3T3=JP2T3+NLT3                                              ID01668Ø
          IPL(JP3T3-2)=IPL(JP2T3-2)                                     ID01669Ø
          IPL(JP3T3-1)=IPL(JP2T3-1)                                     ID01670Ø
          IPL(JP3T3)  =IPL(JP2T3)                                       ID01671Ø
   53   CONTINUE                                                        ID01672Ø
        DO 54  JP2T3=3,NLT3,3                                           ID01673Ø
          JP3T3=JP2T3+NSHT3                                             ID01674Ø
          IPL(JP2T3-2)=IPL(JP3T3-2)                                     ID01675Ø
          IPL(JP2T3-1)=IPL(JP3T3-1)                                     ID01676Ø
          IPL(JP2T3)  =IPL(JP3T3)                                       ID01677Ø
   54   CONTINUE                                                        ID01678Ø
        JPMX=JPMX-NSH                                                   ID01679Ø
C - ADDS TRIANGLES TO THE IPT ARRAY, UPDATES BORDER LINE                ID01680Ø
C - SEGMENTS IN THE IPL ARRAY, AND SETS FLAGS FOR THE BORDER            ID01681Ø
C - LINE SEGMENTS TO BE REEXAMINED IN THE IWL ARRAY.                    ID01682Ø
   6Ø   JWL=Ø                                                           ID01683Ø
```

```
      DO 64  JP2=JPMX,NLØ                                              IDØ1684Ø
        JP2T3=JP2*3                                                    IDØ1685Ø
        IPL1=IPL(JP2T3-2)                                              IDØ1686Ø
        IPL2=IPL(JP2T3-1)                                              IDØ1687Ø
        IT  =IPL(JP2T3)                                                IDØ1688Ø
C - - ADDS A TRIANGLE TO THE IPT ARRAY.                                IDØ1689Ø
        NTØ=NTØ+1                                                      IDØ1690Ø
        NTT3=NTT3+3                                                    IDØ1691Ø
        IPT(NTT3-2)=IPL2                                               IDØ1692Ø
        IPT(NTT3-1)=IPL1                                               IDØ1693Ø
        IPT(NTT3)  =IP1                                                IDØ1694Ø
C - - UPDATES BORDER LINE SEGMENTS IN THE IPL ARRAY.                   IDØ1695Ø
        IF(JP2.NE.JPMX)      GO TO 61                                  IDØ1696Ø
        IPL(JP2T3-1)=IP1                                               IDØ1697Ø
        IPL(JP2T3)  =NTØ                                               IDØ1698Ø
   61   IF(JP2.NE.NLØ)       GO TO 62                                  IDØ1699Ø
        NLN=JPMX+1                                                     IDØ17ØØØ
        NLNT3=NLN*3                                                    IDØ17Ø1Ø
        IPL(NLNT3-2)=IP1                                               IDØ17Ø2Ø
        IPL(NLNT3-1)=IPL(1)                                            IDØ17Ø3Ø
        IPL(NLNT3)  =NTØ                                               IDØ17Ø4Ø
C - - DETERMINES THE VERTEX THAT DOES NOT LIE ON THE BORDER            IDØ17Ø5Ø
C - - LINE SEGMENTS.                                                   IDØ17Ø6Ø
   62   ITT3=IT*3                                                      IDØ17Ø7Ø
        IPTI=IPT(ITT3-2)                                               IDØ17Ø8Ø
        IF(IPTI.NE.IPL1.AND.IPTI.NE.IPL2)   GO TO 63                   IDØ17Ø9Ø
        IPTI=IPT(ITT3-1)                                               IDØ171ØØ
        IF(IPTI.NE.IPL1.AND.IPTI.NE.IPL2)   GO TO 63                   IDØ1711Ø
        IPTI=IPT(ITT3)                                                 IDØ1712Ø
C - - CHECKS IF THE EXCHANGE IS NECESSARY.                             IDØ1713Ø
   63   IF(IDXCHG(XD,YD,IP1,IPTI,IPL1,IPL2).EQ.Ø)     GO TO 64          IDØ1714Ø
C - - MODIFIES THE IPT ARRAY WHEN NECESSARY.                           IDØ1715Ø
        IPT(ITT3-2)=IPTI                                               IDØ1716Ø
        IPT(ITT3-1)=IPL1                                               IDØ1717Ø
        IPT(ITT3)  =IP1                                                IDØ1718Ø
        IPT(NTT3-1)=IPTI                                               IDØ1719Ø
        IF(JP2.EQ.JPMX)      IPL(JP2T3)=IT                             IDØ172ØØ
        IF(JP2.EQ.NLØ.AND.IPL(3).EQ.IT)     IPL(3)=NTØ                IDØ1721Ø
C - - SETS FLAGS IN THE IWL ARRAY.                                     IDØ1722Ø
        JWL=JWL+4                                                      IDØ1723Ø
        IWL(JWL-3)=IPL1                                                IDØ1724Ø
        IWL(JWL-2)=IPTI                                                IDØ1725Ø
        IWL(JWL-1)=IPTI                                                IDØ1726Ø
        IWL(JWL)  =IPL2                                                IDØ1727Ø
   64   CONTINUE                                                       IDØ1728Ø
        NLØ=NLN                                                        IDØ1729Ø
        NLT3=NLNT3                                                     IDØ173ØØ
        NLF=JWL/2                                                      IDØ1731Ø
        IF(NLF.EQ.Ø)      GO TO 79                                     IDØ1732Ø
C - IMPROVES TRIANGULATION.                                            IDØ1733Ø
   7Ø   NTT3P3=NTT3+3                                                  IDØ1734Ø
        DO 78  IREP=1,NREP                                             IDØ1735Ø
          DO 76  ILF=1,NLF                                             IDØ1736Ø
            ILFT2=ILF*2                                                IDØ1737Ø
            IPL1=IWL(ILFT2-1)                                          IDØ1738Ø
            IPL2=IWL(ILFT2)                                            IDØ1739Ø
C - - LOCATES IN THE IPT ARRAY TWO TRIANGLES ON BOTH SIDES OF          IDØ174ØØ
C - - THE FLAGGED LINE SEGMENT.                                        IDØ1741Ø
            NTF=Ø                                                      IDØ1742Ø
            DO 71  ITT3R=3,NTT3,3                                      IDØ1743Ø
              ITT3=NTT3P3-ITT3R                                        IDØ1744Ø
              IPT1=IPT(ITT3-2)                                         IDØ1745Ø
              IPT2=IPT(ITT3-1)                                         IDØ1746Ø
              IPT3=IPT(ITT3)                                           IDØ1747Ø
              IF(IPL1.NE.IPT1.AND.IPL1.NE.IPT2.AND.                    IDØ1748Ø
     1           IPL1.NE.IPT3)      GO TO 71                          IDØ1749Ø
              IF(IPL2.NE.IPT1.AND.IPL2.NE.IPT2.AND.                    IDØ175ØØ
     1           IPL2.NE.IPT3)      GO TO 71                          IDØ1751Ø
              NTF=NTF+1                                                IDØ1752Ø
              ITF(NTF)=ITT3/3                                          IDØ1753Ø
              IF(NTF.EQ.2)     GO TO 72                                IDØ1754Ø
   71       CONTINUE                                                  IDØ1755Ø
            IF(NTF.LT.2)       GO TO 76                              IDØ1756Ø
C - - DETERMINES THE VERTEXES OF THE TRIANGLES THAT DO NOT LIE         IDØ1757Ø
C - - ON THE LINE SEGMENT.                                             IDØ1758Ø
   72       IT1T3=ITF(1)*3                                            IDØ1759Ø
```

```
            IPTI1=IPT(IT1T3-2)                                          ID01760
            IF(IPTI1.NE.IPL1.AND.IPTI1.NE.IPL2)    GO TO 73             ID01761
            IPTI1=IPT(IT1T3-1)                                          ID01762
            IF(IPTI1.NE.IPL1.AND.IPTI1.NE.IPL2)    GO TO 73             ID01763
            IPTI1=IPT(IT1T3)                                            ID01764
   73       IT2T3=ITF(2)*3                                              ID01765
            IPTI2=IPT(IT2T3-2)                                          ID01766
            IF(IPTI2.NE.IPL1.AND.IPTI2.NE.IPL2)    GO TO 74             ID01767
            IPTI2=IPT(IT2T3-1)                                          ID01768
            IF(IPTI2.NE.IPL1.AND.IPTI2.NE.IPL2)    GO TO 74             ID01769
            IPTI2=IPT(IT2T3)                                            ID01770
C - - CHECKS IF THE EXCHANGE IS NECESSARY.                              ID01771
   74       IF(IDXCHG(XD,YD,IPTI1,IPTI2,IPL1,IPL2).EQ.0)                ID01772
     1        GO TO 76                                                  ID01773
C - - MODIFIES THE IPT ARRAY WHEN NECESSARY.                            ID01774
            IPT(IT1T3-2)=IPTI1                                          ID01775
            IPT(IT1T3-1)=IPTI2                                          ID01776
            IPT(IT1T3)  =IPL1                                           ID01777
            IPT(IT2T3-2)=IPTI2                                          ID01778
            IPT(IT2T3-1)=IPTI1                                          ID01779
            IPT(IT2T3)  =IPL2                                           ID01780
C - - SETS NEW FLAGS.                                                   ID01781
            JWL=JWL+8                                                   ID01782
            IWL(JWL-7)=IPL1                                             ID01783
            IWL(JWL-6)=IPTI1                                            ID01784
            IWL(JWL-5)=IPTI1                                            ID01785
            IWL(JWL-4)=IPL2                                             ID01786
            IWL(JWL-3)=IPL2                                             ID01787
            IWL(JWL-2)=IPTI2                                            ID01788
            IWL(JWL-1)=IPTI2                                            ID01789
            IWL(JWL)  =IPL1                                             ID01790
            DO 75  JLT3=3,NLT3,3                                        ID01791
              IPLJ1=IPL(JLT3-2)                                         ID01792
              IPLJ2=IPL(JLT3-1)                                         ID01793
              IF((IPLJ1.EQ.IPL1.AND.IPLJ2.EQ.IPTI2).OR.                 ID01794
     1           (IPLJ2.EQ.IPL1.AND.IPLJ1.EQ.IPTI2))                    ID01795
     2                        IPL(JLT3)=ITF(1)                         ID01796
              IF((IPLJ1.EQ.IPL2.AND.IPLJ2.EQ.IPTI1).OR.                 ID01797
     1           (IPLJ2.EQ.IPL2.AND.IPLJ1.EQ.IPTI1))                    ID01798
     2                        IPL(JLT3)=ITF(2)                         ID01799
   75       CONTINUE                                                    ID01800
   76     CONTINUE                                                      ID01801
          NLFC=NLF                                                      ID01802
          NLF=JWL/2                                                     ID01803
          IF(NLF.EQ.NLFC)       GO TO 79                                ID01804
C - - RESETS THE IWL ARRAY FOR THE NEXT ROUND.                          ID01805
          JWL=0                                                         ID01806
          JWL1MN=(NLFC+1)*2                                             ID01807
          NLFT2=NLF*2                                                   ID01808
          DO 77  JWL1=JWL1MN,NLFT2,2                                    ID01809
            JWL=JWL+2                                                   ID01810
            IWL(JWL-1)=IWL(JWL1-1)                                      ID01811
            IWL(JWL)  =IWL(JWL1)                                        ID01812
   77     CONTINUE                                                      ID01813
          NLF=JWL/2                                                     ID01814
   78   CONTINUE                                                       ID01815
   79 CONTINUE                                                         ID01816
C REARRANGES THE IPT ARRAY SO THAT THE VERTEXES OF EACH TRIANGLE        ID01817
C ARE LISTED COUNTER-CLOCKWISE.                                         ID01818
   80 DO 81  ITT3=3,NTT3,3                                             ID01819
        IP1=IPT(ITT3-2)                                                 ID01820
        IP2=IPT(ITT3-1)                                                 ID01821
        IP3=IPT(ITT3)                                                   ID01822
        IF(SIDE(XD(IP1),YD(IP1),XD(IP2),YD(IP2),XD(IP3),YD(IP3))        ID01823
     1       .GE.0.0)     GO TO 81                                      ID01824
        IPT(ITT3-2)=IP2                                                 ID01825
        IPT(ITT3-1)=IP1                                                 ID01826
   81 CONTINUE                                                          ID01827
      NT=NT0                                                            ID01828
      NL=NL0                                                            ID01829
      RETURN                                                            ID01830
C ERROR EXIT                                                            ID01831
   90 WRITE (LUN,2090)  NDP0                                            ID01832
      GO TO 93                                                          ID01833
   91 WRITE (LUN,2091)  NDP0,IP1,IP2,X1,Y1                              ID01834
      GO TO 93                                                          ID01835
```

```
   92 WRITE (LUN,2Ø92)  NDPØ                                          IDØ1836Ø
   93 WRITE (LUN,2Ø93)                                                IDØ1837Ø
      NT=Ø                                                            IDØ1838Ø
      RETURN                                                          IDØ1839Ø
C FORMAT STATEMENTS                                                   IDØ1840Ø
 2Ø9Ø FORMAT(1X/23H ***   NDP LESS THAN 4./8H   NDP =,I5)            IDØ1841Ø
 2Ø91 FORMAT(1X/29H ***   IDENTICAL DATA POINTS./                     IDØ1842Ø
     1   8H   NDP =,I5,5X,5HIP1 =,I5,5X,5HIP2 =,I5,                  IDØ1843Ø
     2   5X,4HXD =,E12.4,5X,4HYD =,E12.4)                            IDØ1844Ø
 2Ø92 FORMAT(1X/33H ***   ALL COLLINEAR DATA POINTS./                 IDØ1845Ø
     1   8H   NDP =,I5)                                              IDØ1846Ø
 2Ø93 FORMAT(35H ERROR DETECTED IN ROUTINE   IDTANG/)                IDØ1847Ø
      END                                                             IDØ1848Ø


      FUNCTION  IDXCHG(X,Y,I1,I2,I3,I4)                               IDØ1856Ø
C THIS FUNCTION DETERMINES WHETHER OR NOT THE EXCHANGE OF TWO         IDØ1857Ø
C TRIANGLES IS NECESSARY ON THE BASIS OF MAX-MIN-ANGLE CRITERION      IDØ1858Ø
C BY C. L. LAWSON.                                                    IDØ1859Ø
C THE INPUT PARAMETERS ARE                                            IDØ1860Ø
C     X,Y = ARRAYS CONTAINING THE COORDINATES OF THE DATA             IDØ1861Ø
C           POINTS,                                                   IDØ1862Ø
C     I1,I2,I3,I4 = POINT NUMBERS OF FOUR POINTS P1, P2,              IDØ1863Ø
C           P3, AND P4 THAT FORM A QUADRILATERAL WITH P3              IDØ1864Ø
C           AND P4 CONNECTED DIAGONALLY.                              IDØ1865Ø
C THIS FUNCTION RETURNS AN INTEGER VALUE 1 (ONE) WHEN AN EX-          IDØ1866Ø
C CHANGE IS NECESSARY, AND Ø (ZERO) OTHERWISE.                        IDØ1867Ø
C DECLARATION STATEMENTS                                              IDØ1868Ø
      DIMENSION   X(1ØØ),Y(1ØØ)                                       IDØ1869Ø
      EQUIVALENCE (C2SQ,C1SQ),(A3SQ,B2SQ),(B3SQ,A1SQ),                IDØ1870Ø
     1            (A4SQ,B1SQ),(B4SQ,A2SQ),(C4SQ,C3SQ)                IDØ1871Ø
C PRELIMINARY PROCESSING                                              IDØ1872Ø
   1Ø X1=X(I1)                                                        IDØ1873Ø
      Y1=Y(I1)                                                        IDØ1874Ø
      X2=X(I2)                                                        IDØ1875Ø
      Y2=Y(I2)                                                        IDØ1876Ø
      X3=X(I3)                                                        IDØ1877Ø
      Y3=Y(I3)                                                        IDØ1878Ø
      X4=X(I4)                                                        IDØ1879Ø
      Y4=Y(I4)                                                        IDØ1880Ø
C CALCULATION                                                         IDØ1881Ø
   2Ø IDX=Ø                                                           IDØ1882Ø
      U3=(Y2-Y3)*(X1-X3)-(X2-X3)*(Y1-Y3)                              IDØ1883Ø
      U4=(Y1-Y4)*(X2-X4)-(X1-X4)*(Y2-Y4)                              IDØ1884Ø
      IF(U3*U4.LE.Ø.Ø)    GO TO 3Ø                                     IDØ1885Ø
      U1=(Y3-Y1)*(X4-X1)-(X3-X1)*(Y4-Y1)                              IDØ1886Ø
      U2=(Y4-Y2)*(X3-X2)-(X4-X2)*(Y3-Y2)                              IDØ1887Ø
      A1SQ=(X1-X3)**2+(Y1-Y3)**2                                      IDØ1888Ø
      B1SQ=(X4-X1)**2+(Y4-Y1)**2                                      IDØ1889Ø
      C1SQ=(X3-X4)**2+(Y3-Y4)**2                                      IDØ1890Ø
      A2SQ=(X2-X4)**2+(Y2-Y4)**2                                      IDØ1891Ø
      B2SQ=(X3-X2)**2+(Y3-Y2)**2                                      IDØ1892Ø
      C3SQ=(X2-X1)**2+(Y2-Y1)**2                                      IDØ1893Ø
      S1SQ=U1*U1/(C1SQ*AMAX1(A1SQ,B1SQ))                              IDØ1894Ø
      S2SQ=U2*U2/(C2SQ*AMAX1(A2SQ,B2SQ))                              IDØ1895Ø
      S3SQ=U3*U3/(C3SQ*AMAX1(A3SQ,B3SQ))                              IDØ1896Ø
      S4SQ=U4*U4/(C4SQ*AMAX1(A4SQ,B4SQ))                              IDØ1897Ø
      IF(AMIN1(S1SQ,S2SQ).LT.AMIN1(S3SQ,S4SQ))     IDX=1              IDØ1898Ø
   3Ø IDXCHG=IDX                                                      IDØ1899Ø
      RETURN                                                          IDØ19ØØØ
      END                                                             IDØ19Ø1Ø
```

ACM Transactions on Mathematical Software, Vol. 5, No. 2, June 1979, Pages 242-243.

REMARK ON ALGORITHM 526

Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points [E1]
[H. Akima, *ACM Trans. Math. Software 4,* 2 (June 1978), 160-164]

Hiroshi Akima [Recd 19 December 1978]
U.S. Department of Commerce, National Telecommunications and Information Administration, Institute for Telecommunication Sciences, Boulder, CO 80303

In Section 10.2.6, ANSI Standard Fortran provides that the completion of execution of a RETURN statement in a subprogram causes all local variables in that subprogram to become undefined. Some of the subprograms in Algorithm 526, however, are dependent on the retention of values by local variables. The algorithm, therefore, will fail to work properly if the above provision is strictly enforced as in the case where the "reset own" option is used on the Burroughs B6700 computer.

Illegal use of undefined local variables will take place in the following cases:
(1) in IDBVIP when called with MD = 2 or 3;
(2) in IDSFFT when called with MD = 2 or 3;
(3) in IDLCTN when IDBVIP is called with NIP > 1;
(4) in IDPTIP when IDBVIP is called with NIP > 1;
(5) in IDPTIP when IDSFFT is called with NXI > 1 and/or NYI > 1.

Cases (1) and (2) can be corrected by resetting two integer variables in IDBVIP and IDSFFT, respectively. Case (3) can be corrected by combining a local integer variable with another integer variable in a COMMON area shared by IDBVIP and IDLCTN, and by placing 13 real variables in the same COMMON area. Cases (4) and (5) can be corrected by placing 27 real variables in a COMMON area shared by IDBVIP, IDPTIP, and IDSFFT. (Corresponding to the last correction, some additional minor changes are required in IDPTIP.) These corrections require no changes for the program that calls IDBVIP and/or IDSFFT.

Specifically, the following replacement, insertion, and deletion should be made:

```
*************        REQUIRED CHANGES IN ACM ALGORITHM 526       *************

                                  ***   IDBVIP   ***

      COMMON/IDLC/ITIPV,DMMY1(13)                                        ID001880
      COMMON/IDPI/ITPV,DMMY(27)                                          ID001890
   40 IF(MD0.GT.1)   GO TO 41                                            ID002240
      GO TO 50                                                           ID002291
   41 NT=IWK(5)                                                          ID002292
      NL=IWK(6)                                                          ID002293
      ITIPV=0                                                            ID002370

                                  ***   IDLCTN   ***

      DIMENSION   IDSC(9)                                                IDL  370
      COMMON/IDLC/ITIPV,XS1,XS2,YS1,YS2,NTSC(9)                          IDL  380
      IF(ITIPV.NE.0)      GO TO 80                                       IDL  500
C     NIT=1         ***   THIS LINE SHOULD BE DELETED.   ***             IDL  510

                                  ***   IDPTIP   ***

      COMMON/IDPI/ITPV,X0,Y0,AP,BP,CP,DP,                                ID010470
     1            P00,P10,P20,P30,P40,P50,P01,P11,P21,P31,P41,           ID010471
     2            P02,P12,P22,P32,P03,P13,P23,P04,P14,P05                ID010472
      X0=XD(IDP)                                                         ID012750
      Y0=YD(IDP)                                                         ID012760
      Z0=ZD(IDP)                                                         ID012770
   67 P00=Z0                                                             ID012840
   70 U=XII-X0                                                           ID012920
      V=YII-Y0                                                           ID012930
```

```
                                   ***   IDSFFT   ***

      COMMON/IDPI/ITPV,DMMY(27)                                          ID013640
   40 IF(MD0.GT.1)    GO TO 41                                           ID014030
      GO TO 50                                                           ID014081
   41 NT=IWK(5)                                                          ID014082
      NL=IWK(6)                                                          ID014083

*******************************************************************************
```

After these changes are made, the algorithm works properly with the "reset own" option on the B6700 computer.

ACKNOWLEDGMENT

The author is grateful to Lars Mossberg of Volvo Flygmotor AB, Trollhättan, Sweden, for calling attention to this problem and for testing the corrected algorithm.

527-P 1- 0

ALGORITHM 527
A Fortran Implementation of the Generalized Marching Algorithm [D3]

RANDOLPH E. BANK
The University of Chicago

Key Words and Phrases: marching algorithms, block tridiagonal, elliptic partial differential equations
CR Categories: 5.14, 5.17
Language: Fortran

1. DESCRIPTION

Subroutines GMA and GMAS are implementations of the generalized marching algorithm [1, 2], which may be used to solve linear systems arising from 5-point discretizations of separable or constant coefficient elliptic boundary-value problems on rectangular domains. A Dirichlet, Neumann, or mixed boundary condition may be independently specified on each side of the rectangle; periodic boundary conditions may be specified on opposing sides. An appropriate linear system is block tridiagonal, or nearly block tridiagonal, real, symmetric, positive or negative definite (for GMA) or semidefinite (for GMAS), and of the form $Ax = b$:

$$\begin{bmatrix} T+\alpha_1 I & -\beta_2 I & & & -\beta_1 I \\ -\beta_2 I & T+\alpha_2 I & -\beta_3 I & & \\ & \ddots & \ddots & \ddots & \\ & & -\beta_{N-1} I & T+\alpha_{N-1} & -\beta_N I \\ -\beta_1 I & & & -\beta_N I & T+\alpha_N I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{N-1} \\ b_N \end{bmatrix} \tag{1}$$

Here $T = TM$ is the (nearly) tridiagonal $M \times M$ matrix

$$TM = \begin{bmatrix} \gamma_1 & -\sigma_2 & & & -\sigma_r \\ -\sigma_2 & \gamma_2 & -\sigma_3 & & \\ & \ddots & \ddots & \ddots & \\ & & -\sigma_{M-1} & \gamma_{M-1} & -\sigma_M \\ -\sigma_1 & & & -\sigma_M & \gamma_M \end{bmatrix} \tag{2}$$

Received 1 April 1976 and 15 February 1977.

This program was written while the author was a summer visitor at Argonne National Laboratory, Argonne, IL.
Authors' present address: Department of Mathematics, University of Texas, Austin, TX 78712.

The α_i, β_i, γ_i, and σ_i are scalars, and x_i and b_i are M-vectors. Many common discretizations which do not automatically yield symmetric matrices A (for example, in the case of Neumann or mixed boundary conditions) can be transformed into the form of eq. (1) using diagonal similarity transformations [4, 11]. No restrictions on N and M, other than N, M $\geq$ 3 are imposed.

The eigenvalues of certain principal submatrices of the N×N matrix TN, given by

$$\mathrm{TN} = \begin{bmatrix} \alpha_1 & -\beta & & & -\beta_1 \\ -\beta_2 & \alpha_{22} & -\beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & -\beta_{N-1} & \alpha_{N-1} & -\beta_N \\ -\beta_1 & & & -\beta_N & \alpha_N \end{bmatrix} \tag{3}$$

are required by the generalized marching algorithm and must be computed in a separate preprocessing phase. To solve a problem, the user must call GMA or GMAS twice, first to carry out the preprocessing, and then to solve the linear system. (If the same linear system is to be solved with many right-hand sides, the preprocessing need be done only once if the contents of the array R are saved.) Nineteen other subroutines are called internally as required.

PARTN, ROOTSC, ROOTSG, SORT, QL, and BANDR are entered during the preprocessing phase. In PARTN, the matrix A is partitioned into ND + 1 blocks by ND = [N/K] separating lines (the M unknowns in the vector x_J comprise the J-th line). Each block has K or fewer lines, where K is the marching parameter specified by the user or supplied by default. The integer NE, which satisfies $2^{NE-1} \leq$ ND $\leq 2^{NE} - 1$ if ND > 0, NE = 0 if ND = 0, is also computed.

ROOTSC and ROOTSG perform eigenvalue calculations for constant coefficient and general separable problems, respectively. SORT orders the eigenvalues according to size, both to insure numerical stability and to allow certain reductions in computation to occur in the constant coefficient case. QL and BANDR are called by ROOTSC and ROOTSG in connection with solving the eigenvalue problems; they are adaptations of TQLRAT and BANDR from the EISPACK subroutine library [7, 12].

In the backsolution phase, subroutines STEP1, STEP2, STEP3, STEP4, TRI1, TRI2, MARCH1, MARCH2, and MARCH3 are entered. In STEP1 the ND + 1 problems associated with the partitioned matrix A are partially solved using the marching algorithm. Next, in STEP2, NE − 1 "2-reductions" are applied to a reduced set of equations involving the separating lines. In STEP3, the solution vectors x_J corresponding to the separating lines are calculated in NE recursive substeps, effectively decomposing the N line problem into ND + 1 smaller problems. In STEP4, these smaller problems are solved via the marching algorithm. If periodic boundary conditions are specified for the matrix TN, certain adjustments are made in the backsolution process.

In the backsolution phase, O(N·NE) linear systems of the form $(\mathrm{T} - r\mathrm{I})x = y$ are solved; this is done in TRI1 and TRI2. Additionally, the backsolution requires O(N) matrix multiplications of the form $x = \mathrm{T}y$, which are carried out in MARCH1, MARCH2, and MARCH3.

Subroutine GMA is entered using

```
CALL GMA(NPC, N, NTYPE, TN1, TN2, M, MTYPE, TM1, TM2, IBDIM, B, K, R, A)
```

The parameter list is:

NPC is an integer. If NPC = 0, GMA does preprocessing calculations; if NPC = 1 GMA solves the linear system.
N is the dimension of the matrix TN; N > 2.
NTYPE is an integer describing the matrix TN.

NTYPE = 1: General separable; Dirichlet, Neumann, or mixed boundary conditions.
$\beta_1 = 0$; $\beta_i \neq 0$, $2 \leq i \leq N$;
α_i arbitrary, $1 \leq i \leq N$.
NTYPE = 2: Constant coefficient; Dirichlet, Neumann, or mixed boundary conditions.
$\beta_1 = 0$; $\beta_i = \beta \neq 0$, $3 \leq i \leq N - 1$;
$\alpha_i = \alpha$, α arbitrary, $2 \leq i \leq N - 1$.

Top boundary condition: One of

$\beta_2 = \beta$; $\alpha_1 = \alpha$	(Dirichlet)
$\beta_2 = (\sqrt{2})\beta$; $\alpha_1 = \alpha$	(Neumann centered)
$\beta_2 = \beta$; $\alpha_1 = \alpha - \beta$	(Neumann noncentered)
$\beta_2 = \psi \neq 0$; $\alpha_1 = \rho$, ρ arbitrary	(mixed).

Bottom boundary condition: One of

$\beta_N = \beta$; $\alpha_N = \alpha$	(Dirichlet)
$\beta_N = (\sqrt{2})\beta$; $\alpha_N = \alpha$	(Neumann centered)
$\beta_N = \beta$; $\alpha_N = \alpha - \beta$	(Neumann noncentered)
$\beta_N = \chi \neq 0$; $\alpha_N = \eta$, η arbitrary	(mixed).

NTYPE = 3: General separable; periodic boundary conditions.
$\beta_i \neq 0$, $1 \leq i \leq N$;
α_i arbitrary, $1 \leq i \leq N$.
NTYPE = 4: Constant coefficient; periodic boundary conditions.
$\beta_i = \beta \neq 0$, $1 \leq i \leq N$;
$\alpha_i = \alpha$, α arbitrary, $1 \leq i \leq N$.
If N = 3, GMA may respecify NTYPE.

TN1 is a real array of length N + 1, with TN1(I) = α_I, $1 \leq I \leq N$.
TN2 is a real array of length N + 1, with TN2(I) = β_I, $1 \leq I \leq N$.
Input values of TN2(N + 1), and TN2(1) if NTYPE $\leq$ 2, may be overwritten by GMA.
M is the dimension of the matrix TM; M > 2.
MTYPE is an integer describing the matrix TM.
MTYPE = 1: General separable; Dirichlet, Neumann, or mixed boundary conditions.
MTYPE = 2: Constant coefficient; Dirichlet, Neumann, or mixed boundary conditions.
MTYPE = 3: General separable; periodic boundary conditions.
MTYPE = 4: Constant coefficient; periodic boundary conditions.
Restrictions on γ_i and σ_i are analogous to those for α_i and β_i, respectively, for each given type. If M $\leq$ 5, GMA may respecify MTYPE.
TM1 is a real array of length M + 1, with TM1(I) = γ_I, $1 \leq I \leq M$.
TM2 is a real array of length M + 1, with TM2(I) = σ_I, $1 \leq I \leq M$.
IBDIM is the row dimension of the array B, as it appears in the calling program.
B is a two-dimensional array with at least M rows and N columns. On input, B contains the right-hand side, with the vector b_I residing in the I-th column; B(J, I) = $(b_I)_J$, $1 \leq J \leq M$, $1 \leq I \leq N$. On output B contains the solution, with the vector x_I residing in the I-th column; B(J, I) = $(x_I)_J$, $1 \leq J \leq M$, $1 \leq I \leq N$.
K is the marching parameter. If K < 2, then K assumes the default value K = 2.
R is a real array of length N·P + 3·N + 2, where $\log_2 (N/K) \leq P$, P an integer, K $\geq$ 2. The array R contains the output of a call to GMA with NPC = 0 (preprocessing calculations).
A is a real array of length 4·Q, where Q = max(N + 1, M + 1). This array is used as a scratchpad array by GMA.

Subroutine GMA has one labeled common block, /MACHEP/, containing one

variable, TOL. TOL is a machine-dependent constant, which is equal to the machine epsilon. It is initialized in GMA in the first executable statement.

When the matrix A of eq. (1) is positive or negative *semi*definite, with zero an eigenvalue of multiplicity one, then subroutine GMAS should be used in place of GMA to compute a least-squares solution. The usage of GMAS is nearly identical to that of GMA, though the two differ internally in several respects.

First, in the preprocessing phase (NPC = 0), the eigenvector v corresponding to the zero eigenvalue is determined by computing appropriate eigenvectors of the matrices TN and TM. The particular root S, for which TM − SI is singular, is perturbed to guard against possible divide checks.

Second, in the backsolution phase (NPC = 1), the right-hand side b is projected into the orthogonal complement of the vector v. The right-hand sides for the reduced equations after STEP1 and STEP2 are also projected into their appropriate orthogonal complements, since the original orthogonalities may be diminished by roundoff growth during these steps of the backsolution. Finally, the solution x is projected into the orthogonal complement of v. This completes the least-squares solution.

Subroutine GMAS calls internally the 15 subroutines employed by GMA. Additionally, GMAS calls subroutines TINVIT, PINVIT, SVALUE, and PROJ as required. TINVIT and PINVIT are used to compute eigenvectors of symmetric tridiagonal matrices and symmetric matrices with periodic boundary conditions, respectively, using the method of inverse iteration [11]. TINVIT is a modified version of EISPACK subroutine of the same name [7, 12]. Subroutine SVALUE computes the perturbation of the root S. PROJ computes the projections described above.

Subroutine GMAS is entered using

```
CALL GMAS(NPC, N, NTYPE, TN1, TN2, M, MTYPE, TM1, TM2, IBDIM, B, VN, VM
  K, R, A).
```

All parameters except VN, VM, and A are identical to the corresponding parameters in the calling sequence for GMA.

VN is a real array of length N + 1. On output from a call to GMAS with NPC = 0, VN contains the eigenvector of TN associated with the zero eigenvalue.

VM is a real array of length M + 1. On output from a call to GMAS with NPC = 0, VM contains the eigenvector of TM associated with the zero eigenvalue.

A is a real array of length 5·Q, where Q = max(N + 1, M + 1). This array is used as a scratchpad array by GMAS.

2. TESTS

The operation count for subroutine GMA (GMAS) is $O(N \cdot M \cdot NE) = O(N \cdot M \cdot \log_2 (N/K))$, with the constant depending on the problem specification and the boundary conditions. (If NE = 0 this becomes $O(N \cdot M)$). Constant coefficient problems are important special cases of general separable problems in which the operation count can be reduced, often dramatically, by taking advantage of the additional restrictions on the coefficients. In Table I we have assembled execution times (in milliseconds) for GMA on an IBM 370-195 computer for some benchmark values of N = M and K. Both constant coefficient and general separable problems were solved to illustrate the savings which can accrue in the special cases. The programs were compiled using the Fortran-H compiler, option = 2. Overall execution time is given as a sum of preprocessing time (NPC = 0) and backsolution time (NPC = 1).

The constant coefficient problems were all Helmholtz equations in the unit

square, with the Helmholtz constant a pseudorandom number between zero and one. Dirichlet boundary conditions were imposed on all boundaries. For the general separable problems, the β_i and σ_i, $1 \leq i \leq N + 1$, were all pseudorandom numbers between one and two. Then $\alpha_i = \beta_i + \beta_{i+1} + \rho_i$ and $\gamma_i = \sigma_i + \sigma_{i+1} + \eta_i$, $1 \leq i \leq N$, were calculated, with ρ_i and η_i pseudorandom numbers between zero and N^{-2}; this insured that the resulting matrices were diagonally dominant, hence positive definite.

The savings which result in the constant coefficient case depend to some extent

Table I. Execution Times for Subroutine GMA on the IBM 370-195

(P = preprocessing time (NPC = 0) in milliseconds, B = backsolution time (NPC = 1) in milliseconds, T = total execution time in milliseconds. NTYPE = MTYPE = 1 in general separable problems, NTYPE = MTYPE = 2 in constant coefficient problems, N = M in all problems.)

		Constant coefficient problems			General separable problems		
		N = 31	N = 63	N = 127	N = 31	N = 63	N = 127
K = 2							
	P	3	7	17	22	73	245
	B	24	103	449	44	215	1033
	T	27	110	446	66	288	1278
K = 4							
	P	3	6	14	22	72	243
	B	22	99	444	37	191	949
	T	25	105	458	59	263	1192
K = 8							
	P	2	5	12	19	68	234
	B	18	86	401	27	153	806
	T	20	91	413	46	221	1040
K = 16							
	P	1	4	9	15	59	217
	B	14	72	350	17	112	633
	T	15	76	359	32	171	850
K = 32							
	P	1	2	7	9	48	194
	B	8	50	287	9	70	457
	T	9	52	294	18	118	651

on the boundary conditions, the Dirichlet problem illustrated being the optimal case. For both constant coefficient and general separable problems, the execution times increase if periodic boundary conditions are imposed.

In Table II we present some results illustrating the numerical stability of the generalized marching algorithm as a function of N = M and K for each problem type. An exact solution x_e of pseudorandom numbers on $[-1, 1]$ was generated and substituted into the difference equation in order to generate a right-hand side. The linear system was then solved, and the computed solution x_c was compared with the exact solution. The number of correct digits was computed using digits $= -\log_{10}((e^T e / x_e^T x_e)^{1/2})$, $e = x_e - x_c$. REAL*8 arithmetic was used (approximately 16 decimal digits).

For K = 2, 4, the error primarily reflects the condition number of the matrix A. When K = 8, 16, 32, however, an exponential "marching" term becomes the dominant term in the error [1].

Note that both execution time and numerical stability decrease with increasing K; thus one must seek to strike a balance between rapid execution times and accuracy requirements.

Table II. Numerical Stability of Subroutine GMA on the IBM 370-195

(NTYPE = MTYPE = 1 in general separable problems, NTYPE = MTYPE = 2 in constant coefficient problems, N = M in all problems.)

	constant coefficient problems			general separable problems		
	N = 31	N = 63	N = 127	N = 31	N = 63	N = 127
K = 2	14.8	14.3	13.9	13.3	12.5	12.6
K = 4	14.2	14.0	13.9	13.3	12.5	12.6
K = 8	11.1	11.1	11.0	11.1	11.0	11.6
K = 16	4.9	4.9	4.9	4.6	4.1	4.0
K = 32	0.0	0.0	0.0	0.0	0.0	0.0

Subroutine KPICK is designed to aid the user in making appropriate choices of the marching parameter K and to provide other information useful in determining if a linear system is appropriate for GMA or GMAS.

KPICK can be used in two ways: first, to find an optimal value of K for a user-specified accuracy demand, or, second, to estimate the accuracy for a user-selected value of K.

KPICK is based on a heuristic which supposes that subroutine GMA (or GMAS) yields solutions satisfying error estimates of the form (COND + EMARCH)*TOL, where COND is the condition number of the linear system EMARCH is the marching error, and TOL is the machine epsilon. In [1], error bounds of this form are proved for constant coefficient Dirichlet problems when $N = K2^L - 1$ for a non-negative integer L.

Subroutine KPICK is entered using

CALL KPICK(NPC, N, NTYPE, TN1, TN2, M, MTYPE, TM1, TM2, DEMAND, K, COND, EMARCH, DIGITS, IFLAG)

The parameter list is:

NPC is an integer. If NPC = 0, KPICK determines K such that the solutions computed using subroutine GMA or GMAS will have approximately DEMAND significant digits. If NPC = 1, subroutine will test the input value of K.

N, NTYPE, TN1, TN2, M, MTYPE, TM1, TM2 are identical to the corresponding parameters in the calling sequence for subroutine GMA.

DEMAND is the real number, stating the number of significant digits desired in the computed solution. It must be specified if NPC = 0.

K is an integer. If NPC = 0, on output K is equal to the marching parameter determined by KPICK. If NPC = 1, the user must specify the value of K to be tested as an input parameter.

COND is a real number, normally returning the condition number of the linear system.

EMARCH is a real number, normally returning an estimate of the marching error for the output value of K.

DIGITS is a real number, normally returning a value of max(0.0, −log((COND + EMARCH)*TOL). This is an estimate of the number of significant digits, in the 2-norm, one may expect in the computed solution.

IFLAG is an integer, describing error returns.

IFLAG = 0: Normal return.

IFLAG = 1: KPICK failed to successfully compute COND. The linear system is not positive or negative definite. If the linear system is positive or negative semidefinite with a zero eigenvalue of multiplicity one, the condition is computed

relative to the orthogonal complement of the eigenvector associated with the zero eigenvalue. Otherwise the default values COND = EMARCH = DIGITS = 0.0 are returned.

IFLAG = 2: The marching error may not satisfy the assumptions underlying the algorithm used to compute EMARCH. (The marching error appears to grow more slowly than $(2 \cdot 5)^K$.) If NPC = 0, KPICK returns estimates for K = 2; if NPC = 1, KPICK returns estimates for the input value of K.

IFLAG = 3: Conditions 1 and 2 exist.

IFLAG = 4: The linear system cannot be solved to DEMAND significant digits for any $K \geq 2$. KPICK returns estimates for K = 2. This return can occur only if NPC = 0.

IFLAG = 5: Conditions 1 and 4 exist.

IFLAG = 6: Conditions 2 and 4 exist.

IFLAG = 7: Conditions 1, 2, and 4 exist.

IFLAG = 8: The input values of N, NTYPE, TN1, and/or TN2 are incorrectly specified.

IFLAG = 9: The input values of M, MTYPE, TM1, and/or TM2 are incorrectly specified.

IFLAG = 10: Conditions 8 and 9 exist.

If IFLAG = 0, 2, 4, 6, then the problem is appropriate for subroutine GMA with the qualifications noted above. If IFLAG = 1, 3, 5, 7, with other than default values for COND, EMARCH, and DIGITS, then the problem is appropriate for GMAS.

Subroutine KPICK calls subroutines TCHECK, EIGEN, TRIEIG, PEREIG, and ERROR as required. TCHECK is used to determine if the matrices TN and TM have been correctly specified. Then COND is determined by computing the extremal eigenvalues of TN and TM; this is done in EIGEN. For some matrix types, the eigenvalues are well known; otherwise TRIEIG or PEREIG are used. These routines compute eigenvalues of symmetric tridiagonal and periodic matrices, respectively, using bisection and the Sturm sequence property [4, 11].

For a fixed value of K, EMARCH is defined to be the largest 2-norm over the set of marching polynomials arising in STEP1 or STEP4 of the generalized marching algorithm [1, 2]. For a fixed value of K, subroutine ERROR is used to compute the marching error; the three-term recurrence relation for generating the marching polynomials is employed [1, 2]. If NPC = 1 computation of EMARCH is straightforward; if NPC = 0, the method of bisection is used to determine the optimal value of K, and hence EMARCH.

To test the effectiveness of KPICK, 3000 random problems (1000 nonsingular, 1000 singular, and 1000 where singularity was randomly determined) were solved on an IBM 370-195 using REAL*8 arithmetic. All required input values for KPICK, GMA, and GMAS were generated using random number generators of the type used in testing the EISPACK codes [7].

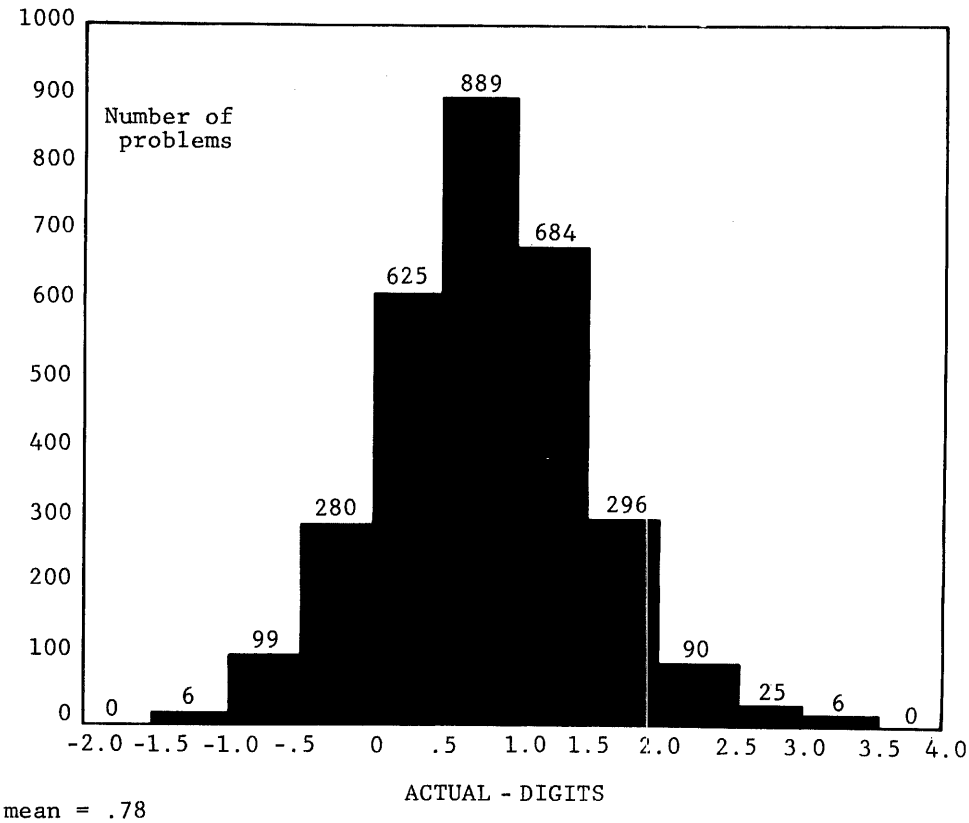
The values of N and M were random integers between 3 and 63; NTYPE and MTYPE were random integers between 1 and 4 (for type-2 matrices, the boundary condition combination was determined randomly). The matrix entries themselves were generated using random real numbers on $[10^{-3}, 1.0]$. Positive or negative definiteness (semidefiniteness) was determined randomly.

The parameter DEMAND was a random real number on $[1.0, -\log(\text{TOL}) - 1.0]$. Exact solutions were generated using random numbers on $[-1.0, 1.0]$ and were substituted into the difference equations to generate right-hand sides.

The random problems were analyzed by KPICK, with NPC = 0; the values of K produced in this fashion ranged between 2 and 29. The problems were then solved, using GMA or GMAS, and the computed solutions compared with the exact solutions. The number of correct digits, in the 2-norm, of the computed solution ACTUAL was determined and compared with the estimate of DIGITS provided by KPICK. The results are summarized in Table III, where we have

Table III. Effectiveness of Subroutine KPICK in Predicting the Error for 3000 Random Problems

(ACTUAL = actual number of correct digits in the computed solution in the 2-norm using GMA or GMAS, DIGITS = number of correct digits predicted by KPICK.)



mean = .78

variance = .48

graphed the value of ACTUAL–DIGITS versus the number of problems. The results indicate the effectiveness of KPICK in predicting the accuracy of computed solutions using GMA and GMAS. The results for singular and nonsingular problems considered separately are essentially the same as those given in Table III.

Other Fortran programs for solving eq. (1) or certain special cases using fast direct methods can be found in [3, 8]; theoretical discussions of the algorithms are given in [5, 6, 9, 10].

ACKNOWLEDGMENT

The author is indebted to Paul Concus of Lawrence Berkeley Laboratory, whose tests and critiques of preliminary versions of the program led to many improvements in the final version, and to Jack Dongarra of Argonne National Laboratory for many helpful discussions.

REFERENCES

1. Bank, R.E., and Rose, D.J. Marching algorithms for elliptic boundary value problems I: The constant coefficient case. *SIAM J. Numer. Anal. 14* (1977), 792–829.
2. Bank, R.E. Marching algorithms for elliptic boundary value problems II: The variable coefficient case. *SIAM J. Numer. Anal. 14* (1977), 950–970.
3. Bank, R.E. FORTRAN implementations of marching algorithms. Tech. Rep. TR 17-75, Ctr. for Res. in Comptng. Tech., Harvard U., Cambridge, Mass., May 1975.
4. Björck, A., and Golub, G.H. Eigenproblems for matrices associated with periodic boundary conditions. *SIAM Rev. 19* (1977), 5–16
5. Buzbee, B.L., Golub, G.H., and Neilson, C.W. On direct methods of solving Poisson's equation. *SIAM J. Numer. Anal. 7* (1970), 627–656.

6. DORR, F.W. The direct solution of the discrete Poisson equation on a rectangle. *SIAM Rev. 12* (1970), 248-263.
7. SMITH, B.T., ET AL. *Matrix Eigensystem Routines*. Springer-Verlag, New York, 1974.
8. SWARZTRAUBER, P.N., AND SWEET, R.A. Efficient FORTRAN subprograms for the solution of elliptic partial differential equations. Tech. Note NCAR-TN/IA-109, Atmospheric Tech. Div., Nat. Ctr. for Atmospheric Res., Boulder, Colo., July 1975.
9. SWARZTRAUBER, P.N. A direct method for the discrete solution of separable elliptic problems. *SIAM J. Numer. Anal. 11* (1974), 1136-1150.
10. SWEET, R.A. A generalized cyclic reduction algorithm. *SIAM J. Numer. Anal. 11* (1974), 506-520.
11. WILKINSON, J.H. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, 1965.
12. WILKINSON, J.H., AND REINSCH, C. *Handbook for Automatic Computation, Vol. II: Linear Algebra*. Springer-Verlag, New York, 1971.

ALGORITHM

```
      SUBROUTINE GMA(NPC,N,NTYPE,TN1,TN2,M,MTYPE,                       GMAØØØ1Ø
     1     TM1,TM2,IBDIM,B,K,R,A)                                       GMAØØØ2Ø
C         SUBROUTINE GMA IS AN IMPLEMENTATION OF THE GENERALIZED        GMAØØØ3Ø
C       MARCHING ALGORITHM, WHICH MAY BE USED FOR SOLVING LINEAR        GMAØØØ4Ø
C       SYSTEMS ARISING FROM 5 POINT DISCRETIZATIONS OF SEPARABLE       GMAØØØ5Ø
C       OR CONSTANT COEFFICIENT ELLIPTIC BOUNDARY VALUE PROBLEMS        GMAØØØ6Ø
      ON RECTANGULAR DOMAINS.  A DIRICHLET, NEUMANN, OR MIXED           GMAØØØ7Ø
      BOUNDARY CONDITION MAY BE INDEPENDENTLY SPECIFIED ON EACH         GMAØØØ8Ø
      SIDE OF THE RECTANGLE, OR PERIODIC BOUNDARY CONDITIONS MAY        GMAØØØ9Ø
      BE SPECIFIED ON OPPOSING SIDES.                                   GMAØØ1ØØ
          THE LINEAR SYSTEM HAS THE FORM:                               GMAØØ11Ø
                                                                        GMAØØ12Ø
      ( TM1(I) + TN1(J) ) * X(I,J)                                      GMAØØ13Ø
          - TM2(I+1) * X(I+1,J)  -  TM2(I) * X(I-1,J)                   GMAØØ14Ø
          - TN2(J+1) * X(I,J+1)  -  TN2(J) * X(I,J-1)  =  B(I,J)        GMAØØ15Ø
                                                                        GMAØØ16Ø
      FOR I = 1,2,...,M, AND J = 1,2,...,N, WHERE TM2(M+1) IS           GMAØØ17Ø
      INTERPRETED AS TM2(1),  TN2(N+1) AS TN2(1),  X(Ø,J) AS X(M,J),    GMAØØ18Ø
      X(M+1,J) AS X(1,J),  X(I,Ø) AS X(I,N),  AND X(I,N+1) AS X(I,1).   GMAØØ19Ø
                                                                        GMAØØ2ØØ
      THE PARAMETER LIST:                                               GMAØØ21Ø
                                                                        GMAØØ22Ø
      NPC IS AN INTEGER.  IF NPC = Ø, GMA DOES NECESSARY                GMAØØ23Ø
          PREPROCESSING CALCULATIONS.  IF NPC = 1, GMA SOLVES THE       GMAØØ24Ø
          LINEAR SYSTEM.                                                GMAØØ25Ø
      N IS AN INTEGER, AS DEFINED ABOVE. N MUST BE GREATER THAN 2.      GMAØØ26Ø
      NTYPE IS AN INTEGER DESCRIBING THE ARRAYS TN1 AND TN2.            GMAØØ27Ø
          NTYPE = 1: GENERAL SEPARABLE - DIRICHLET, NEUMANN, OR         GMAØØ28Ø
          MIXED BOUNDARY CONDITIONS.                                    GMAØØ29Ø
          TN1(I) IS ARBITRARY, I = 1,2,...,N.                           GMAØØ3ØØ
          TN2(1) = Ø.Ø; TN2(I) IS ARBITRARY, NONZERO, I = 2,3,...,N.    GMAØØ31Ø
          NTYPE = 2: CONSTANT COEFFICIENT - DIRICHLET, NEUMANN,         GMAØØ32Ø
          OR MIXED BOUNDARY CONDITIONS.                                 GMAØØ33Ø
          TN1(I) = ALPHA, ALPHA AN ARBITRARY CONSTANT, I = 2,3,...N-1.  GMAØØ34Ø
          TN2(1) = Ø.Ø; TN2(I) = BETA, BETA AN ARBITRARY NONZERO        GMAØØ35Ø
          CONSTANT, I = 3,...,N-1.                                      GMAØØ36Ø
          TOP BOUNDARY CONDITION : ONE OF                               GMAØØ37Ø
          TN1(1) = ALPHA;  TN2(2) = BETA; (DIRICHLET)                   GMAØØ38Ø
          TN1(1) = ALPHA;  TN2(2) = SQRT(2) * BETA; (NEUMANN-CENTERED)  GMAØØ39Ø
          TN1(1) = ALPHA - BETA;  TN2(2) = BETA;  (NEUMANN-STAGGERED)   GMAØØ4ØØ
          TN1(1) = RHO, RHO ARBITRARY; TN2(2) = ZETA, ZETA ARBITRARY,   GMAØØ41Ø
          NONZERO; (MIXED).                                             GMAØØ42Ø
          BOTTOM BOUNDARY CONDITION : ONE OF                            GMAØØ43Ø
          TN1(N) = ALPHA;  TN2(N) = BETA; (DIRICHLET)                   GMAØØ44Ø
          TN1(N) = ALPHA;  TN2(N) = SQRT(2) * BETA; (NEUMANN-CENTERED)  GMAØØ45Ø
          TN1(N) = ALPHA - BETA;  TN2(N) = BETA;  (NEUMANN-STAGGERED)   GMAØØ46Ø
          TN1(N) = CHI, CHI ARBITRARY; TN2(N) = ETA, ETA ARBITRARY,     GMAØØ47Ø
          NONZERO; (MIXED).                                             GMAØØ48Ø
          NTYPE = 3: GENERAL SEPARABLE - PERIODIC BOUNDARY CONDITIONS.  GMAØØ49Ø
          TN1(I) IS ARBITRARY, I = 1,2,...,N.                           GMAØØ5ØØ
          TN2(I) IS ARBITRARY, NONZERO, I = 1,2,...,N.                  GMAØØ51Ø
          NTYPE = 4: CONSTANT COEFFICIENT - PERIODIC BOUNDARY           GMAØØ52Ø
          CONDITIONS.                                                   GMAØØ53Ø
          TN1(I) = ALPHA, ALPHA ARBITRARY, I = 1,2,...,N.               GMAØØ54Ø
          TN2(I) = BETA, BETA ARBITRARY, NONZERO, I = 1,2,...,N.        GMAØØ55Ø
          IF N = 3, GMA MAY RESPECIFY NTYPE.                            GMAØØ56Ø
      TN1 IS AN ARRARY OF LENGTH N+1, DEFINED ABOVE.                    GMAØØ57Ø
      TN2 IS AN ARRAY OF LENGTH N+1, DEFINED ABOVE.                     GMAØØ58Ø
      M IS AN INTEGER, AS DEFINED ABOVE. M MUST BE GREATER THAN 2.      GMAØØ59Ø
```

```
C       MTYPE IS AN INTEGER DESCRIBING THE ARRAYS TM1 AND TM2.          GMAØØ6ØØ
C           MTYPE = 1: GENERAL SEPARABLE - DIRICHLET, NEUMANN, OR       GMAØØ61Ø
C           MIXED BOUNDARY CONDITIONS.                                  GMAØØ62Ø
C           MTYPE = 2: CONSTANT COEFFICIENT - DIRICHLET, NEUMANN,       GMAØØ63Ø
C           OR MIXED BOUNDARY CONDITIONS.                               GMAØØ64Ø
C           MTYPE = 3: GENERAL SEPARABLE - PERIODIC BOUNDARY CONDITIONS.GMAØØ65Ø
C           MTYPE = 4: CONSTANT COEFFICIENT - PERIODIC BOUNDARY         GMAØØ66Ø
C           CONDITIONS.                                                 GMAØØ67Ø
C           RESTRICTIONS ON TM1 AND TM2 ARE ANALOGOUS TO THOSE ON       GMAØØ68Ø
C           TN1 AND TN2, RESPECTIVELY, FOR EACH GIVEN TYPE.             GMAØØ69Ø
C           IF M IS LESS THAN 6, GMA MAY RESPECIFY MTYPE.               GMAØØ7ØØ
C       TM1 IS AN ARRAY OF LENGTH M+1, DEFINED ABOVE.                   GMAØØ71Ø
C       TM2 IS AN ARRAY OF LENGTH M+1, DEFINED ABOVE.                   GMAØØ72Ø
C       IBDIM IS THE ROW DIMENSION OF THE ARRAY B, AS IT APPEARS        GMAØØ73Ø
C           IN THE CALLING PROGRAM.                                     GMAØØ74Ø
C       B IS A TWO DIMENSIONAL ARRAY WITH AT LEAST M ROWS AND N         GMAØØ75Ø
C           COLUMNS. ON INPUT, B CONTAINS THE RIGHT HAND SIDE B(I,J)    GMAØØ76Ø
C           AS DEFINED ABOVE.  ON OUTPUT, FROM A CALL TO GMA WITH       GMAØØ77Ø
C           NPC = 1, B CONTAINS THE SOLUTION X(I,J), AS DEFINED ABOVE.  GMAØØ78Ø
C       K IS THE MARCHING PARAMETER.  MARCHING OCCURS IN THE            GMAØØ79Ø
C           N - DIRECTION.  ON INPUT, IF K IS LESS THAN 2, IT           GMAØØ8ØØ
C           ASSUMES THE DEFAULT VALUE K = 2.                            GMAØØ81Ø
C       R IS AN ARRAY OF LENGTH N * P + 3 * N + 2, WHERE P IS AN        GMAØØ82Ø
C           INTEGER GREATER THAN OR EQUAL TO LOG2( N / K ), AND K IS    GMAØØ83Ø
C           GREATER THAN OR EQUAL TO 2.  R CONTAINS THE OUTPUT FROM A   GMAØØ84Ø
C           CALL TO GMA WITH NPC = Ø.                                   GMAØØ85Ø
C       A IS AN ARRAY OF LENGTH 4 * Q, WHERE Q = MAX( N+1, M+1 ).       GMAØØ86Ø
C           A IS USED AS A SCRATCHPAD ARRAY BY GMA.                     GMAØØ87Ø
C                                                                       GMAØØ88Ø
C           SUBROUTINE GMA HAS ONE LABELED COMMON BLOCK, /MACHEP/,      GMAØØ89Ø
C       WITH ONE VARIABLE, TOL.  TOL IS A MACHINE DEPENDENT CONSTANT,   GMAØØ9ØØ
C       EQUAL TO THE MACHINE EPSILON.  IT IS INITIALIZED IN GMA IN THE  GMAØØ91Ø
C       FIRST EXECUTABLE STATEMENT.                                     GMAØØ92Ø
C                                                                       GMAØØ93Ø
C           A CALL TO GMA WITH NPC = Ø MUST PRECEDE  A CALL TO GMA      GMAØØ94Ø
C       WITH NPC = 1.  IF THE SAME LINEAR SYSTEM IS TO BE SOLVED        GMAØØ95Ø
C       WITH SEVERAL RIGHT HAND SIDES, A SINGLE CALL TO GMA WITH        GMAØØ96Ø
C       NPC = Ø WILL SUFFICE, PROVIDED THAT THE CONTENTS OF THE         GMAØØ97Ø
C       ARRAY R ARE SAVED.                                              GMAØØ98Ø
C           SUBROUTINE GMA MAY FAIL TO GIVE CORRECT ANSWERS IF          GMAØØ99Ø
C       THE LINEAR SYSTEM IS NOT POSITIVE OR NEGATIVE DEFINITE.         GMAØ1ØØØ
C       IF N, NTYPE, M, MTYPE, OR IBDIM ARE OUT OF RANGE, GMA RETURNS   GMAØ1Ø1Ø
C       WITHOUT ATTEMPTING ANY COMPUTATIONS.                            GMAØ1Ø2Ø
C                                                                       GMAØ1Ø3Ø
C       ADDRESS INQUIRIES TO:                                           GMAØ1Ø4Ø
C           RANDOLPH E. BANK                                            GMAØ1Ø5Ø
C           DEPARTMENT OF MATHEMATICS                                   GMAØ1Ø6Ø
C           THE UNIVERSITY OF CHICAGO                                   GMAØ1Ø7Ø
C           CHICAGO ILLINOIS 6Ø637.                                     GMAØ1Ø8Ø
C                                                                       GMAØ1Ø9Ø
C       VERSION DATE: MARCH 1, 1977.                                    GMAØ11ØØ
            DIMENSION TN1(1),TN2(1),TM1(1),TM2(1),A(1),B(1),R(1)        GMAØ111Ø
            COMMON /MACHEP/TOL                                          GMAØ112Ø
        TOL=2.ØEØ**(-23)                                                GMAØ113Ø
C       CHECK N, NTYPE, M, MTYPE, AND IBDIM.                            GMAØ114Ø
        IF((NTYPE.GT.4).OR.(NTYPE.LT.1)) RETURN                         GMAØ115Ø
        IF((MTYPE.GT.4).OR.(MTYPE.LT.1)) RETURN                         GMAØ116Ø
        IF((N.LT.2).OR.(M.LT.2)) RETURN                                 GMAØ117Ø
        IF(IBDIM.LT.M) RETURN                                           GMAØ118Ø
C       RESPECIFY NTYPE AND/OR MTYPE IF NECESSARY.                      GMAØ119Ø
        IF(NTYPE.EQ.2.AND.N.LE.3) NTYPE=1                               GMAØ12ØØ
        IF(MTYPE.EQ.2.AND.M.LE.3) MTYPE=1                               GMAØ121Ø
        IF(NTYPE.EQ.4.AND.N.LE.3) NTYPE=3                               GMAØ122Ø
        IF(MTYPE.EQ.4.AND.M.LE.5) MTYPE=3                               GMAØ123Ø
        IF(N.LE.2) NTYPE=1                                              GMAØ124Ø
        IF(M.LE.2) MTYPE=1                                              GMAØ125Ø
        IF(NPC.NE.Ø) GO TO 15                                           GMAØ126Ø
C       THE PREPROCESSING CALCULATIONS.                                 GMAØ127Ø
        CALL PARTN(N,NTYPE,K,NE,ND,R(3))                                GMAØ128Ø
C       EPS IS ADDED TO ALL INTEGERS STORED IN THE REAL ARRAY R.        GMAØ129Ø
C       THIS PREVENTS UNFORTUNATE ROUNDING ERRORS FORM OCCURRING WHEN   GMAØ13ØØ
C       THEY ARE CONVERTED BACK INTO INTEGERS.                          GMAØ131Ø
        EPS=Ø.25EØ                                                      GMAØ132Ø
        R(1)=FLOAT(NE)+EPS                                              GMAØ133Ø
        R(2)=FLOAT(ND)+EPS                                              GMAØ134Ø
        IF(NTYPE.EQ.1.OR.NTYPE.EQ.3) CALL ROOTSG(N,NTYPE,               GMAØ135Ø
     1      TN1,TN2,NE,ND,R(3),R(N+3),A)                                GMAØ136Ø
        IF(NTYPE.EQ.2.OR.NTYPE.EQ.4) CALL ROOTSC(N,NTYPE,               GMAØ137Ø
```

```
     1      TN1,TN2,NE,ND,R(3),R(N+3),A)                                  GMAØ138Ø
            J1=N+2                                                        GMAØ139Ø
            J2=J1+N+1                                                     GMAØ14ØØ
            CALL SORT(N,NTYPE,NE,ND,R(3),R(N+3),A(1),A(J1),A(J2))         GMAØ141Ø
            RETURN                                                        GMAØ142Ø
C           THE BACKSOLUTION CALCULATIONS.                                GMAØ143Ø
    15      TN2(N+1)=1.ØEØ                                                GMAØ144Ø
            IF(NTYPE.LT.3) TN2(1)=1.ØEØ                                   GMAØ145Ø
            NE=R(1)                                                       GMAØ146Ø
            ND=R(2)                                                       GMAØ147Ø
            CALL STEP1(N,NTYPE,TN1,TN2,M,MTYPE,TM1,TM2,                   GMAØ148Ø
     1          IBDIM,B,ND,R(3),R(N+3),A)                                GMAØ149Ø
            CALL STEP2(N,NTYPE,TN2,M,MTYPE,TM1,TM2,                       GMAØ15ØØ
     1          IBDIM,B,NE,ND,R(3),R(N+3),A)                             GMAØ151Ø
            CALL STEP3(N,NTYPE,TN2,M,MTYPE,TM1,TM2,                       GMAØ152Ø
     1          IBDIM,B,NE,ND,R(3),R(N+3),A)                             GMAØ153Ø
            CALL STEP4(TN1,TN2,M,MTYPE,TM1,TM2,IBDIM,B,ND,R(3),R(N+3),A)  GMAØ154Ø
            RETURN                                                        GMAØ155Ø
            END                                                           GMAØ156Ø
C
C
            SUBROUTINE PARTN(N,NTYPE,K,NE,ND,ID)                          PRTNØØ1Ø
C               SUBROUTINE PARTN PARTITIONS THE GRID, IN THE N - DIRECTION, PRTNØØ2Ø
C           INTO ND+1 BLOCKS, EACH WITH K OR FEWER LINES.  POINTERS TO THE PRTNØØ3Ø
C           SEPARATING LINES ARE STORED IN THE ARRAY ID.  THE VALUE       PRTNØØ4Ø
C           OF THE INTEGER NE, ROUGHLY EQUAL TO LOG2(ND), IS COMPUTED.    PRTNØØ5Ø
C           EPS IS ADDED TO ALL INTEGERS STORED IN THE REAL ARRAY ID.     PRTNØØ6Ø
C           THIS PREVENTS UNFORTUNATE ROUNDING ERRORS FORM OCCURRING WHEN PRTNØØ7Ø
C           THEY ARE CONVERTED BACK INTO INTEGERS.                        PRTNØØ8Ø
C           VERSION DATE: FEBRUARY 1, 1977.                               PRTNØØ9Ø
                REAL ID                                                   PRTNØ1ØØ
                DIMENSION ID(1)                                           PRTNØ11Ø
            EPS=Ø.25EØ                                                    PRTNØ12Ø
            NN=N                                                          PRTNØ13Ø
            IF(NTYPE.GE.3) NN=N-1                                         PRTNØ14Ø
            ND=NN/MAXØ(K,2)                                               PRTNØ15Ø
            NE=Ø                                                          PRTNØ16Ø
            IF(2*ND.EQ.NN) ND=ND-1                                        PRTNØ17Ø
            IF(ND.EQ.Ø) GO TO 1Ø                                          PRTNØ18Ø
            I1=NN/(ND+1)                                                  PRTNØ19Ø
            I2=NN-I1*(ND+1)                                               PRTNØ2ØØ
            I3=Ø                                                          PRTNØ21Ø
            DO 5 J=1,ND                                                   PRTNØ22Ø
                IF(I3.LT.I2) I3=I3+1                                      PRTNØ23Ø
                J1=J*I1+I3                                                PRTNØ24Ø
                ID(J+1)=FLOAT(J1)+EPS                                     PRTNØ25Ø
                J2=2**(J-1)                                               PRTNØ26Ø
                IF(J2.LE.ND.AND.ND.LE.(2*J2-1)) NE=J                      PRTNØ27Ø
     5          CONTINUE                                                 PRTNØ28Ø
    1Ø      ID(1)=EPS                                                     PRTNØ29Ø
            I1=ND+2                                                       PRTNØ3ØØ
            DO 15 J=I1,N                                                  PRTNØ31Ø
    15          ID(J)=FLOAT(NN+1)+EPS                                     PRTNØ32Ø
            RETURN                                                        PRTNØ33Ø
            END                                                           PRTNØ34Ø
C
C
            SUBROUTINE ROOTSC(N,NTYPE,TN1,TN2,NE,ND,ID,R,A)               RTSCØØ1Ø
C               SUBROUTINE ROOTSC DOES THE NECESSARY EIGENVALUE           RTSCØØ2Ø
C           CALCULATIONS FOR CONSTANT COEFFICIENT PROBLEMS.               RTSCØØ3Ø
C           THE EIGENVALUES ARE STORED IN THE ARRAY R.                    RTSCØØ4Ø
C           VERSION DATE: FEBRUARY 1, 1977.                               RTSCØØ5Ø
                REAL ID                                                   RTSCØØ6Ø
                DIMENSION TN1(1),TN2(1),ID(1),R(1),A(1)                   RTSCØØ7Ø
            NP1=N+1                                                       RTSCØØ8Ø
            NEP1=NE+1                                                     RTSCØØ9Ø
            NDP1=ND+1                                                     RTSCØ1ØØ
            PI=3.14159265358979323846264338327 95EØ                       RTSCØ11Ø
            T1=TN1(2)                                                     RTSCØ12Ø
            T2=TN2(3)                                                     RTSCØ13Ø
            T22=2.ØEØ*T2                                                  RTSCØ14Ø
            S=ABS(T22)                                                    RTSCØ15Ø
C           IN THE FOLLOWING SEGMENT OF CODE, SPECIAL EIGENVALUES FOR THE RTSCØ16Ø
C           CASE NTYPE = 4 ARE COMPUTED.                                  RTSCØ17Ø
            IF(NTYPE.EQ.2) GO TO 13Ø                                      RTSCØ18Ø
            LINE=N*NEP1                                                   RTSCØ19Ø
            I2=LINE+N                                                     RTSCØ2ØØ
```

```
      R(I2)=T1-S                                                        RTSCØ21Ø
      R(LINE+1)=T1+S                                                    RTSCØ22Ø
      NM1=N-1                                                           RTSCØ23Ø
      I2=2                                                              RTSCØ24Ø
      Q=PI/FLOAT(N)                                                     RTSCØ25Ø
      IF(T22.LT.Ø.ØEØ) GO TO 11Ø                                        RTSCØ26Ø
      LINE=LINE+N                                                       RTSCØ27Ø
      I2=-2                                                             RTSCØ28Ø
 11Ø  DO 115 I=2,NM1,2                                                  RTSCØ29Ø
          LINE=LINE+I2                                                  RTSCØ3ØØ
          ARG=Q*FLOAT(I)                                                RTSCØ31Ø
          R(LINE)=T1-T22*COS(ARG)                                       RTSCØ32Ø
 115      R(LINE+1)=R(LINE)                                             RTSCØ33Ø
C     IN THE FOLLOWING SEGMENT OF CODE, EIGENVALUES RELEVANT TO ALL     RTSCØ34Ø
C     CONSTANT COEFFICIENT PROBLEMS ARE CALCULATED.                     RTSCØ35Ø
 13Ø  DO 14Ø I=1,NEP1                                                   RTSCØ36Ø
          I2=2**(I-1)                                                   RTSCØ37Ø
          J2=N*(I-1)                                                    RTSCØ38Ø
      DO 14Ø LNINDX=1,NDP1,I2                                           RTSCØ39Ø
          LINE=ID(LNINDX)                                               RTSCØ4ØØ
          ISPAN=LNINDX+I2                                               RTSCØ41Ø
          ISPAN=ID(ISPAN)                                               RTSCØ42Ø
          ISPAN=ISPAN-LINE-1                                            RTSCØ43Ø
          LINE=LINE+J2                                                  RTSCØ44Ø
          Q=PI/FLOAT(ISPAN+1)                                           RTSCØ45Ø
          DO 135 J=1,ISPAN                                              RTSCØ46Ø
              LINE=LINE+1                                               RTSCØ47Ø
              ARG=Q*FLOAT(J)                                            RTSCØ48Ø
 135          R(LINE)=T1+S*COS(ARG)                                     RTSCØ49Ø
          R(LINE+1)=Ø.ØEØ                                               RTSCØ5ØØ
 14Ø      CONTINUE                                                      RTSCØ51Ø
      IF(NTYPE.EQ.4) RETURN                                             RTSCØ52Ø
C     IN THE FOLLOWING SEGMENT OF CODE, WE CHECK FOR NEUMANN OR MIXED   RTSCØ53Ø
C     BOUNDARY CONDITIONS, AND MODIFY SOME EIGENVALUES IN THE           RTSCØ54Ø
C     APPROPRIATE FASHION.                                              RTSCØ55Ø
C     ICOUNT KEEPS TRACK OF THE BOUNDARY CONDITION COMBINATION.         RTSCØ56Ø
      ICOUNT=Ø                                                          RTSCØ57Ø
      IA1=1                                                             RTSCØ58Ø
      IA2=Ø                                                             RTSCØ59Ø
      S2=SQRT(2.ØEØ)*T2                                                 RTSCØ6ØØ
      DO 27Ø L=1,2                                                      RTSCØ61Ø
          I2=2*IA1+N*IA2                                                RTSCØ62Ø
          T2P=TN2(I2)                                                   RTSCØ63Ø
          I2=I2-IA1                                                     RTSCØ64Ø
          T1P=TN1(I2)                                                   RTSCØ65Ø
          IF(T2P.EQ.T2.AND.T1P.EQ.T1) GO TO 265                         RTSCØ66Ø
          NN=ID(ND+2)                                                   RTSCØ67Ø
          NN=NN*IA2                                                     RTSCØ68Ø
          IF(T1P.EQ.T1.AND.T2P.EQ.S2) GO TO 22Ø                         RTSCØ69Ø
          IF(T2P.EQ.T2.AND.T1P.EQ.(T1-T2)) GO TO 225                    RTSCØ7ØØ
C     THE LOOP FOR A MIXED BOUNDARY CONDITION                           RTSCØ71Ø
          ICOUNT=ICOUNT+7                                               RTSCØ72Ø
          IF(NE.EQ.Ø) GO TO 265                                         RTSCØ73Ø
          DO 215 I=1,NE                                                 RTSCØ74Ø
              I2=2**(I-1)                                               RTSCØ75Ø
              LINE=(ND/I2)*IA2*I2                                       RTSCØ76Ø
              LINE=ID(LINE+1)                                           RTSCØ77Ø
              ISPAN=ID(I2+1)                                            RTSCØ78Ø
              ISPAN=ISPAN*IA1+NN-LINE-1                                 RTSCØ79Ø
              LINE=LINE+N*(I-1)                                         RTSCØ8ØØ
              J2=N                                                      RTSCØ81Ø
              DO 2Ø5 J=1,ISPAN                                          RTSCØ82Ø
                  J2=J2+1                                               RTSCØ83Ø
                  A(J)=T1                                               RTSCØ84Ø
 2Ø5              A(J2)=-T2                                             RTSCØ85Ø
              A(J2)=Ø.ØEØ                                               RTSCØ86Ø
              A(1)=T1P                                                  RTSCØ87Ø
              A(NP1)=-T2P                                               RTSCØ88Ø
              CALL QL(A(1),A(NP1),ISPAN,IERR)                           RTSCØ89Ø
              J2=ISPAN+1                                                RTSCØ9ØØ
              DO 21Ø J=1,ISPAN                                          RTSCØ91Ø
                  J2=J2-1                                               RTSCØ92Ø
                  LINE=LINE+1                                           RTSCØ93Ø
 21Ø              R(LINE)=A(J2)                                         RTSCØ94Ø
 215      CONTINUE                                                      RTSCØ95Ø
          GO TO 265                                                     RTSCØ96Ø
C     THE LOOP FOR A NEUMANN BOUNDARY CONDITION                         RTSCØ97Ø
 22Ø      ICOUNT=ICOUNT+1                                               RTSCØ98Ø
```

```
            INS=Ø                                                    RTSCØ99Ø
            GO TO 23Ø                                                RTSC1ØØØ
  225       ICOUNT=ICOUNT+3                                          RTSC1Ø1Ø
            INS=1                                                    RTSC1Ø2Ø
  23Ø       IF(NE.EQ.Ø) GO TO 265                                    RTSC1Ø3Ø
            J1=Ø                                                     RTSC1Ø4Ø
            J2=2                                                     RTSC1Ø5Ø
            IF(T22.LT.Ø.ØEØ) GO TO 24Ø                               RTSC1Ø6Ø
            J1=2                                                     RTSC1Ø7Ø
            J2=-2                                                    RTSC1Ø8Ø
  24Ø       DO 25Ø I=1,NE                                            RTSC1Ø9Ø
                I2=2**(I-1)                                          RTSC11ØØ
                LINE=(ND/I2)*IA2*I2                                  RTSC111Ø
                LINE=ID(LINE+1)                                      RTSC112Ø
                ISPAN=ID(I2+1)                                       RTSC113Ø
                ISPAN=ISPAN*IA1+NN-LINE-1                            RTSC114Ø
                LINE=LINE+N*(I-1)                                    RTSC115Ø
                Q=PI/FLOAT(2*ISPAN+INS)                              RTSC116Ø
                J=J1*(ISPAN+1)-1                                     RTSC117Ø
                DO 245 JJ=1,ISPAN                                    RTSC118Ø
                    J=J+J2                                           RTSC119Ø
                    LINE=LINE+1                                      RTSC12ØØ
                    ARG=Q*FLOAT(J)                                   RTSC121Ø
  245               R(LINE)=T1-T22*COS(ARG)                          RTSC122Ø
  25Ø           CONTINUE                                             RTSC123Ø
  265           IA1=Ø                                                RTSC124Ø
  27Ø           IA2=1                                                RTSC125Ø
            IF(ICOUNT.EQ.Ø) RETURN                                   RTSC126Ø
            LINE=N*NE                                                RTSC127Ø
            IF(ICOUNT.GE.7) GO TO 4ØØ                                RTSC128Ø
            GO TO (425,43Ø,435,44Ø,4ØØ,445),ICOUNT                   RTSC129Ø
C           IN THE FOLLOWING SEGMENT, EIGENVALUES FOR THE CASE OF ONE RTSC13ØØ
C           OR MORE MIXED BOUNDARY CONDITIONS ARE CALCULATED.        RTSC131Ø
  4ØØ       J2=N                                                     RTSC132Ø
            DO  4Ø5 J=1,N                                            RTSC133Ø
                J2=J2+1                                              RTSC134Ø
                A(J)=T1                                              RTSC135Ø
  4Ø5           A(J2)=-T2                                            RTSC136Ø
            A(1)=TN1(1)                                              RTSC137Ø
            A(N)=TN1(N)                                              RTSC138Ø
            A(NP1)=-TN2(2)                                           RTSC139Ø
            A(J2-1)=-TN2(N)                                          RTSC14ØØ
            A(J2)=Ø.ØEØ                                              RTSC141Ø
            CALL QL(A(1),A(NP1),N,IERR)                              RTSC142Ø
            J2=NP1                                                   RTSC143Ø
            DO 41Ø J=1,N                                             RTSC144Ø
                LINE=LINE+1                                          RTSC145Ø
                J2=J2-1                                              RTSC146Ø
  41Ø           R(LINE)=A(J2)                                        RTSC147Ø
            RETURN                                                   RTSC148Ø
C           IN THE FOLLOWING SEGMENT, EIGENVALUES FOR THE CASE OF ONE RTSC149Ø
C           NEUMANN AND ONE DIRICHLET BOUNDARY CONDITION ARE CALCULATED. RTSC15ØØ
  425       Q=PI/FLOAT(2*N)                                          RTSC151Ø
            T22=-S                                                   RTSC152Ø
            J=-1                                                     RTSC153Ø
            J2=2                                                     RTSC154Ø
            GO TO 45Ø                                                RTSC155Ø
C           IN THE FOLLOWING SEGMENT, EIGENVALUES FOR THE CASE OF TWO RTSC156Ø
C           NEUMANN BOUNDARY CONDITIONS ARE CALCULATED.              RTSC157Ø
  43Ø       Q=PI/FLOAT(N-1)                                          RTSC158Ø
            T22=-S                                                   RTSC159Ø
            J=-1                                                     RTSC16ØØ
            J2=1                                                     RTSC161Ø
            GO TO 45Ø                                                RTSC162Ø
C           IN THE FOLLOWING SEGMENT, EIGENVALUES FOR THE CASE OF ONE RTSC163Ø
C           STAGGERED AND ONE DIRICHLET BOUNDARY CONDITION ARE CALCULATED. RTSC164Ø
  435       Q=PI/FLOAT(2*N+1)                                        RTSC165Ø
            J=-1                                                     RTSC166Ø
            J2=2                                                     RTSC167Ø
            IF(T22.LT.Ø.ØEØ) GO TO 45Ø                               RTSC168Ø
            J=2*N+1                                                  RTSC169Ø
            J2=-2                                                    RTSC17ØØ
            GO TO 45Ø                                                RTSC171Ø
C           IN THE FOLLOWING SEGMENT, EIGENVALUES FOR THE CASE OF ONE RTSC172Ø
C           NEUMANN AND ONE STAGGERED BOUNDARY CONDITION ARE CALCULATED. RTSC173Ø
  44Ø       Q=PI/FLOAT(2*N-1)                                        RTSC174Ø
```

```
      J=-2                                                          RTSC175Ø
      J2=2                                                          RTSC176Ø
      IF(T22.LT.Ø.ØEØ) GO TO 45Ø                                    RTSC177Ø
      J=2*N                                                         RTSC178Ø
      J2=-2                                                         RTSC179Ø
      GO TO 45Ø                                                     RTSC18ØØ
C     IN THE FOLLOWING SEGMENT, EIGENVALUES FOR THE CASE OF TWO     RTSC181Ø
C     STAGGERED BOUNDARY CONDITIONS ARE CALCULATED.                 RTSC182Ø
  445 Q=PI/FLOAT(N)                                                 RTSC183Ø
      J=-1                                                          RTSC184Ø
      J2=1                                                          RTSC185Ø
      IF(T22.LT.Ø.ØEØ) GO TO 45Ø                                    RTSC186Ø
      J=N                                                           RTSC187Ø
      J2=-1                                                         RTSC188Ø
  45Ø DO 455 JJ=1,N                                                 RTSC189Ø
          J=J+J2                                                    RTSC19ØØ
          LINE=LINE+1                                               RTSC191Ø
          ARG=Q*FLOAT(J)                                            RTSC192Ø
  455     R(LINE)=T1-T22*COS(ARG)                                   RTSC193Ø
      RETURN                                                        RTSC194Ø
      END                                                           RTSC195Ø
C
C
      SUBROUTINE ROOTSG(N,NTYPE,TN1,TN2,NE,ND,ID,R,A)               RTSGØØ1Ø
C         SUBROUTINE ROOTSG DOES THE NECESSARY EIGENVALUE           RTSGØØ2Ø
C     CALCULATIONS FOR GENERAL SEPARABLE PROBLEMS.                  RTSGØØ3Ø
C     THE EIGENVALUES ARE STORED ON THE ARRAY R.                    RTSGØØ4Ø
C     VERSION DATE: FEBRUARY 1, 1977.                               RTSGØØ5Ø
          REAL ID                                                   RTSGØØ6Ø
          DIMENSION TN1(1),TN2(1),ID(1),R(1),A(1)                   RTSGØØ7Ø
      NP1=N+1                                                       RTSGØØ8Ø
      NEP1=NE+1                                                     RTSGØØ9Ø
      NDP1=ND+1                                                     RTSGØ1ØØ
C     IN THE FOLLOWING SEGMENT OF CODE, SPECIAL EIGENVALUES FOR THE RTSGØ11Ø
C     CASE NTYPE = 3 ARE COMPUTED.                                  RTSGØ12Ø
      IF(NTYPE.EQ.1) GO TO 5Ø                                       RTSGØ13Ø
      N2=(NP1)/2                                                    RTSGØ14Ø
      NP2=NP1+NP1                                                   RTSGØ15Ø
      NP3=NP1+NP2                                                   RTSGØ16Ø
      DO 35 I=1,N2                                                  RTSGØ17Ø
          II=2*I                                                    RTSGØ18Ø
          I2=NP1-I                                                  RTSGØ19Ø
          I1=NP3+II                                                 RTSGØ2ØØ
          A(I1-1)=TN1(I)                                            RTSGØ21Ø
          A(I1)=TN1(I2)                                             RTSGØ22Ø
          I1=NP2+II                                                 RTSGØ23Ø
          A(I1-1)=Ø.ØEØ                                             RTSGØ24Ø
          A(I1)=Ø.ØEØ                                               RTSGØ25Ø
          I1=NP1+II                                                 RTSGØ26Ø
          A(I1-1)=-TN2(I)                                           RTSGØ27Ø
   35     A(I1)=-TN2(I2+1)                                          RTSGØ28Ø
      A(NP2+2)=-TN2(1)                                              RTSGØ29Ø
      A(NP3-1)=-TN2(N2+1)                                           RTSGØ3ØØ
      A(NP1+1)=Ø.ØEØ                                                RTSGØ31Ø
      A(NP1+2)=Ø.ØEØ                                                RTSGØ32Ø
      CALL BANDR(A(1),A(NP1+1),A(NP2+1),A(NP3+1),N)                 RTSGØ33Ø
      CALL QL(A(1),A(NP1+1),N,IERR)                                 RTSGØ34Ø
      LINE=N*NEP1                                                   RTSGØ35Ø
      J2=NP1                                                        RTSGØ36Ø
      DO 4Ø I=1,N                                                   RTSGØ37Ø
          LINE=LINE+1                                               RTSGØ38Ø
          J2=J2-1                                                   RTSGØ39Ø
   4Ø     R(LINE)=A(J2)                                             RTSGØ4ØØ
C     IN THE FOLLOWING SEGMENT OF CODE, EIGENVALUES RELEVANT TO ALL RTSGØ41Ø
C     GENERAL SEPARABLE PROBLEMS ARE CALCULATED.                    RTSGØ42Ø
   5Ø DO 65 I=1,NEP1                                                RTSGØ43Ø
          I2=2**(I-1)                                               RTSGØ44Ø
          J2=N*(I-1)                                                RTSGØ45Ø
      DO 65 LNINDX=1,NDP1,I2                                        RTSGØ46Ø
          LINE=ID(LNINDX)                                           RTSGØ47Ø
          ISPAN=LNINDX+I2                                           RTSGØ48Ø
          ISPAN=ID(ISPAN)                                           RTSGØ49Ø
          ISPAN=ISPAN-LINE-1                                        RTSGØ5ØØ
          J1=N                                                      RTSGØ51Ø
          I1=LINE                                                   RTSGØ52Ø
          DO 55 J=1,ISPAN                                           RTSGØ53Ø
```

```
                J1=J1+1                                                 RTSG054Ø
                I1=I1+1                                                 RTSG055Ø
                A(J)=TN1(I1)                                            RTSG056Ø
   55           A(J1)=-TN2(I1+1)                                        RTSG057Ø
            A(J1)=Ø.ØEØ                                                 RTSG058Ø
            CALL QL(A(1),A(NP1),ISPAN,IERR)                             RTSG059Ø
            LINE=LINE+J2                                                RTSG06ØØ
            J1=ISPAN+1                                                  RTSG061Ø
            DO 6Ø J=1,ISPAN                                             RTSG062Ø
                LINE=LINE+1                                             RTSG063Ø
                J1=J1-1                                                 RTSG064Ø
   6Ø           R(LINE)=A(J1)                                           RTSG065Ø
            R(LINE+1)=Ø.ØEØ                                             RTSG066Ø
   65       CONTINUE                                                    RTSG067Ø
        RETURN                                                          RTSG068Ø
        END                                                             RTSG069Ø
C                                                                       RTSG07ØØ
C
C
C
        SUBROUTINE SORT(N,NTYPE,NE,ND,ID,R,A,IPI,KPI)                   SORTØØ1Ø
C           SUBROUTINE SORT ORDERS THE EIGENVALUES COMPUTED IN ROOTSC   SORTØØ2Ø
C       OR ROOTSG ACCORDING TO SIZE.  THIS MUST BE DONE TO INSURE THE   SORTØØ3Ø
C       NUMERICAL STABILITY OF THE ALGORITHM.  IN THE CASE OF CONSTANT  SORTØØ4Ø
C       COEFFICIENT PROBLEMS, SORTING THE EIGENVALUES ALLOWS SOME       SORTØØ5Ø
C       REDUCTION IN COMPUTIONS TO TAKE PLACE.                          SORTØØ6Ø
C       THE ARRAY IPI KEEPS TRACK OF THE PERMUTATIONS.                  SORTØØ7Ø
C       VERSION DATE: FEBRUARY 1, 1977.                                 SORTØØ8Ø
            REAL ID,IPI,KPI                                             SORTØØ9Ø
            DIMENSION IPI(1),KPI(1),A(1),R(1),ID(1)                     SORTØ1ØØ
        IF(NE.EQ.Ø) RETURN                                              SORTØ11Ø
        NP1=N+1                                                         SORTØ12Ø
        NN=NP1                                                          SORTØ13Ø
        IF(NTYPE.GE.3) NN=N                                             SORTØ14Ø
        LINET=N*NE+1                                                    SORTØ15Ø
        LINEB=LINET+NN-2                                                SORTØ16Ø
        RMIN=AMIN1(R(LINET),R(LINEB))-2.ØEØ                             SORTØ17Ø
C       EPS IS ADDED TO ALL INTEGERS STORED IN THE REAL ARRAY IPI.      SORTØ18Ø
C       THIS PREVENTS UNFORTUNATE ROUNDING ERRORS FORM OCCURRING WHEN   SORTØ19Ø
C       THEY ARE CONVERTED BACK INTO INTEGERS.                          SORTØ2ØØ
        EPS=Ø.25EØ                                                      SORTØ21Ø
        DO 5Ø5 I=1,N                                                    SORTØ22Ø
            A(I)=R(I)                                                   SORTØ23Ø
            IPI(I)=FLOAT(I)+EPS                                         SORTØ24Ø
  5Ø5       KPI(I)=IPI(I)                                               SORTØ25Ø
        DO 53Ø I=1,NE                                                   SORTØ26Ø
            I2=2**I                                                     SORTØ27Ø
            I1=I2/2                                                     SORTØ28Ø
            DO 51Ø LNINDX=I1,ND,I1                                      SORTØ29Ø
                LINE=ID(LNINDX+1)                                       SORTØ3ØØ
  51Ø           A(LINE)=RMIN                                            SORTØ31Ø
            A(NN)=RMIN                                                  SORTØ32Ø
            DO 52Ø LNINDX=I1,ND,I2                                      SORTØ33Ø
                LINET=LNINDX-I1+1                                       SORTØ34Ø
                LINET=ID(LINET)                                         SORTØ35Ø
                LINET=LINET+1                                           SORTØ36Ø
                LINEB=ID(LNINDX+1)                                      SORTØ37Ø
                LINEB=LINEB+1                                           SORTØ38Ø
                LINE=LINET+1                                            SORTØ39Ø
                ISPAN=LNINDX+I1+1                                       SORTØ4ØØ
                ISPAN=ID(ISPAN)                                         SORTØ41Ø
                ISPAN=ISPAN-1                                           SORTØ42Ø
                KPI(LINET)=IPI(LINEB-1)                                 SORTØ43Ø
            DO 52Ø J=LINE,ISPAN                                         SORTØ44Ø
                IF(A(LINET).GT.A(LINEB)) GO TO 515                      SORTØ45Ø
                KPI(J)=IPI(LINEB)                                       SORTØ46Ø
                LINEB=LINEB+1                                           SORTØ47Ø
                GO TO 52Ø                                               SORTØ48Ø
  515           KPI(J)=IPI(LINET)                                       SORTØ49Ø
                LINET=LINET+1                                           SORTØ5ØØ
  52Ø       CONTINUE                                                    SORTØ51Ø
            LINE=N*I                                                    SORTØ52Ø
            JPI=LINE                                                    SORTØ53Ø
            DO 525 J=1,N                                                SORTØ54Ø
                IPI(J)=KPI(J)                                           SORTØ55Ø
                JPI=JPI+1                                               SORTØ56Ø
  525           A(J)=R(JPI)                                             SORTØ57Ø
```

```
            DO 53Ø J=1,N                                                SORTØ58Ø
                JPI=IPI(J)                                              SORTØ59Ø
                JPI=JPI+LINE                                            SORTØ6ØØ
  53Ø           R(JPI)=A(J)                                             SORTØ61Ø
C       THE FOLLOWING SEGMENT OF CODE SORTS THE SPECIAL EIGENVALUES     SORTØ62Ø
C       WHICH ARISE IN THE CASE OF PERIODIC BOUNDARY CONDITIONS.        SORTØ63Ø
        IF(NTYPE.LT.3) RETURN                                           SORTØ64Ø
        LINE=N*(NE+1)                                                   SORTØ65Ø
        JPI=LINE                                                        SORTØ66Ø
        DO 54Ø J=1,N                                                    SORTØ67Ø
            JPI=JPI+1                                                   SORTØ68Ø
  54Ø       A(J)=R(JPI)                                                 SORTØ69Ø
        DO 545 J=1,N                                                    SORTØ7ØØ
            JPI=IPI(J)                                                  SORTØ71Ø
            JPI=JPI+LINE                                                SORTØ72Ø
  545       R(JPI)=A(J)                                                 SORTØ73Ø
        RETURN                                                          SORTØ74Ø
        END                                                             SORTØ75Ø
C
C
        SUBROUTINE QL(D,E,N,IERR)                                       QLØØØØ1Ø
C           SUBROUTINE QL FINDS ALL THE EIGENVALUES OF A SYMMETRIC      QLØØØØ2Ø
C       TRIDIAGONAL MATRIX, THE RATIONAL (SQUARE ROOT FREE) QL          QLØØØØ3Ø
C       ALGORITHM IS USED.  QL IS CALLED IN CONNECTION WITH FINDING     QLØØØØ4Ø
C       EIGENVALUES FOR GENERAL SEPARABLE PROBLEMS, AND CONSTANT        QLØØØØ5Ø
C       COEFFICIENT PROBLEMS WHERE ONE OR MORE MIXED BOUNDARY CONDITIONSQLØØØØ6Ø
C       ARE PRESENT.  SUBROUTINE QL IS AN ADAPTATION OF SUBROUTINE      QLØØØØ7Ø
C       TQLRAT FROM THE EISPACK SUBROUTINE LIBRARY.                     QLØØØØ8Ø
C       THE DIAGONAL OF THE TRIDIAGONAL MATRIX IS STORED IN THE ARRAY   QLØØØØ9Ø
C       D, THE CODIAGONAL IN E.  THE EIGENVALUES ARE WRITTEN IN D.      QLØØØ1ØØ
C       VERSION DATE: MARCH 1, 1977.                                    QLØØØ11Ø
            COMMON /MACHEP/TOL                                          QLØØØ12Ø
            DIMENSION D(1),E(1)                                         QLØØØ13Ø
        IERR=Ø                                                          QLØØØ14Ø
        IF(N.EQ.1) RETURN                                               QLØØØ15Ø
        NM1=N-1                                                         QLØØØ16Ø
        DO 1ØØ I=1,NM1                                                  QLØØØ17Ø
  1ØØ       E(I) =E(I)* E(I)                                            QLØØØ18Ø
        F=Ø.ØEØ                                                         QLØØØ19Ø
        B=Ø.ØEØ                                                         QLØØØ2ØØ
        C=Ø.ØEØ                                                         QLØØØ21Ø
        E(N)=Ø.ØEØ                                                      QLØØØ22Ø
        DO 29Ø L=1, N                                                   QLØØØ23Ø
            J=Ø                                                         QLØØØ24Ø
            H=TOL*(ABS(D(L))+SQRT(E(L)))                                QLØØØ25Ø
            IF(B.GT.H) GO TO 1Ø5                                        QLØØØ26Ø
            B=H                                                         QLØØØ27Ø
            C=B*B                                                       QLØØØ28Ø
  1Ø5       DO 11Ø M=L, N                                               QLØØØ29Ø
                IF(E(M).LE.C) GO TO 12Ø                                 QLØØØ3ØØ
  11Ø           CONTINUE                                                QLØØØ31Ø
            M=N                                                         QLØØØ32Ø
  12Ø       IF(M.EQ.L) GO TO 21Ø                                        QLØØØ33Ø
  13Ø       IF(J.EQ.3Ø) GO TO 1ØØØ                                      QLØØØ34Ø
            J=J+1                                                       QLØØØ35Ø
            L1=L+1                                                      QLØØØ36Ø
            S=SQRT(E(L))                                                QLØØØ37Ø
            G=D(L)                                                      QLØØØ38Ø
            P=(D(L1)-G)/(2.ØEØ*S)                                       QLØØØ39Ø
            R=SQRT(P*P+1.ØEØ)                                           QLØØØ4ØØ
            D(L)=S/(P+SIGN(R,P))                                        QLØØØ41Ø
            H=G-D(L)                                                    QLØØØ42Ø
            DO 14Ø I=L1, N                                              QLØØØ43Ø
  14Ø           D(I)=D(I)-H                                             QLØØØ44Ø
            F=F+H                                                       QLØØØ45Ø
            G=D(M)                                                      QLØØØ46Ø
            IF(G.EQ.Ø.ØEØ) G=B                                          QLØØØ47Ø
            H=G                                                         QLØØØ48Ø
            S=Ø.ØEØ                                                     QLØØØ49Ø
            MML=M-L                                                     QLØØØ5ØØ
            DO 2ØØ II=1, MML                                            QLØØØ51Ø
                I=M-II                                                  QLØØØ52Ø
                P=G*H                                                   QLØØØ53Ø
                R=P+E(I)                                                QLØØØ54Ø
                E(I+1)=S*R                                              QLØØØ55Ø
                S=E(I)/R                                                QLØØØ56Ø
```

```
                D(I+1)=H+S*(H+D(I))                                    QLØØØ57Ø
                G=D(I)-E(I)/G                                          QLØØØ58Ø
                IF(G.EQ.Ø.ØEØ) G=B                                     QLØØØ59Ø
                H=G*P/R                                                QLØØØ6ØØ
  2ØØ           CONTINUE                                               QLØØØ61Ø
              E(L)=S*G                                                 QLØØØ62Ø
              D(L)=H                                                   QLØØØ63Ø
              IF(H.EQ.Ø.ØEØ) GO TO 21Ø                                 QLØØØ64Ø
              IF(ABS(E(L)).LE.ABS(C/H)) GO TO 21Ø                      QLØØØ65Ø
              E(L)=H*E(L)                                              QLØØØ66Ø
              IF(E(L).NE.Ø.ØEØ) GO TO 13Ø                              QLØØØ67Ø
  21Ø         P=D(L)+F                                                 QLØØØ68Ø
              IF(L.EQ.1) GO TO 25Ø                                     QLØØØ69Ø
              DO 23Ø II=2, L                                           QLØØØ7ØØ
                   I=L+2-II                                            QLØØØ71Ø
                   IF(P.GE.D(I-1)) GO TO 27Ø                           QLØØØ72Ø
                   D(I)=D(I-1)                                         QLØØØ73Ø
  23Ø              CONTINUE                                            QLØØØ74Ø
  25Ø         I=1                                                      QLØØØ75Ø
  27Ø         D(I)=P                                                   QLØØØ76Ø
  29Ø         CONTINUE                                                 QLØØØ77Ø
            GO TO 1ØØ1                                                 QLØØØ78Ø
 1ØØØ       IERR=L                                                     QLØØØ79Ø
 1ØØ1       RETURN                                                     QLØØØ8ØØ
            END                                                        QLØØØ81Ø
C
C
            SUBROUTINE BANDR(D,E,W,V,N)                                BNDRØØ1Ø
C               SUBROUTINE BANDR IS USED TO REDUCE A SYMMETRIC PENTA-  BNDRØØ2Ø
C           DIAGONAL MATRIX TO A SYMMETRIC TRIDIAGONAL MATRIX. BANDR IS BNDRØØ3Ø
C           ENTERED ONLY IN THE CASE NTYPE = 3, AND IS CALLED IN       BNDRØØ4Ø
C           CONNECTION WITH FINDING THE SPECIAL EIGENVALUES ASSOCIATED BNDRØØ5Ø
C           WITH THE PERIODIC BOUNDARY CONDITIONS.  BANDR IS AN ADAPTATION BNDRØØ6Ø
C           OF SUBROUTINE BANDR FROM THE EISPACK SUBROUTINE LIBRARY.   BNDRØØ7Ø
C           THE DIAGONAL OF THE PENTADIAGONAL MATRIX IS STORED IN V,   BNDRØØ8Ø
C           THE FIRST CO-DIAGONAL IN W, AND THE SECOND CO-DIAGONAL IN E. BNDRØØ9Ø
C           THE DIAGONAL OF THE TRIDIAGONAL MATRIX IS WRITTEN IN D, AND BNDRØ1ØØ
C           THE CO-DIAGONAL IS WRITTEN IN E.                           BNDRØ11Ø
C           VERSION DATE: FEBRUARY 1, 1977.                            BNDRØ12Ø
                DIMENSION D(1),E(1),V(1),W(1)                          BNDRØ13Ø
            DO 3Ø J=1, N                                               BNDRØ14Ø
   3Ø           D(J)=1.ØEØ                                             BNDRØ15Ø
            NM2=N-2                                                    BNDRØ16Ø
            DO 7ØØ K=1, NM2                                            BNDRØ17Ø
                KP2=K+2                                                BNDRØ18Ø
                G=E(KP2)                                               BNDRØ19Ø
                E(K+1)=W(K+1)                                          BNDRØ2ØØ
                DO 5ØØ J=KP2, N, 2                                     BNDRØ21Ø
                    JM1=J-1                                            BNDRØ22Ø
                    JP1=J+1                                            BNDRØ23Ø
                    JP2=J+2                                            BNDRØ24Ø
                    IF (G.EQ.Ø.ØEØ) GO TO 7ØØ                          BNDRØ25Ø
                    B1=E(JM1)/G                                        BNDRØ26Ø
                    B2=B1*D(JM1)/D(J)                                  BNDRØ27Ø
                    S2=1.ØEØ/(1.ØEØ+B1*B2)                             BNDRØ28Ø
                    IF (S2.GE.Ø.5EØ ) GO TO 45Ø                        BNDRØ29Ø
                    B1=G/E(JM1)                                        BNDRØ3ØØ
                    B2=B1*D(J)/D(JM1)                                  BNDRØ31Ø
                    C2=1.ØEØ-S2                                        BNDRØ32Ø
                    D(JM1)=C2*D(JM1)                                   BNDRØ33Ø
                    D(J)=C2*D(J)                                       BNDRØ34Ø
                    F1=2.ØEØ*W(J)                                      BNDRØ35Ø
                    F2=B1*V(JM1)                                       BNDRØ36Ø
                    W(J)=-B2*(B1*W(J)-V(J))-F2+W(J)                    BNDRØ37Ø
                    V(JM1)=B2*(B2*V(J)+F1)+V(JM1)                      BNDRØ38Ø
                    V(J)=B1*(F2-F1)+V(J)                               BNDRØ39Ø
                    U=W(JM1)+B2*E(J)                                   BNDRØ4ØØ
                    E(J)=-B1*W(JM1)+E(J)                               BNDRØ41Ø
                    W(JM1)=U                                           BNDRØ42Ø
                    E(JM1)=E(JM1)+B2*G                                 BNDRØ43Ø
                    IF (J.EQ.N) GO TO 5ØØ                              BNDRØ44Ø
                    U=E(JP1)+B2*W(JP1)                                 BNDRØ45Ø
                    W(JP1)=-B1*E(JP1)+W(JP1)                           BNDRØ46Ø
                    E(JP1)=U                                           BNDRØ47Ø
                    IF (JP2.GT.N) GO TO 5ØØ                            BNDRØ48Ø
                    G=B2*E(JP2)                                        BNDRØ49Ø
```

```
                GO TO 5ØØ                                                BNDRØ5ØØ
 45Ø            U=D(JM1)                                                 BNDRØ51Ø
                D(JM1)=S2*D(J)                                           BNDRØ52Ø
                D(J)=S2*U                                                BNDRØ53Ø
                F1=2.ØEØ*W(J)                                            BNDRØ54Ø
                F2=B1*V(J)                                               BNDRØ55Ø
                U=B1*(F2-F1)+V(JM1)                                      BNDRØ56Ø
                W(J)=B2*(B1*W(J)-V(JM1))+F2-W(J)                         BNDRØ57Ø
                V(JM1)=B2*(B2*V(JM1)+F1)+V(J)                            BNDRØ58Ø
                V(J)=U                                                   BNDRØ59Ø
                U=B2*W(JM1)+E(J)                                         BNDRØ6ØØ
                E(J)=-W(JM1)+B1*E(J)                                     BNDRØ61Ø
                W(JM1)=U                                                 BNDRØ62Ø
                E(JM1)=B2*E(JM1)+G                                       BNDRØ63Ø
                IF (J.EQ.N) GO TO 5ØØ                                    BNDRØ64Ø
                U=B2*E(JP1)+W(JP1)                                       BNDRØ65Ø
                W(JP1)=-E(JP1)+B1*W(JP1)                                 BNDRØ66Ø
                E(JP1)=U                                                 BNDRØ67Ø
                IF (JP2.GT.N) GO TO 5ØØ                                  BNDRØ68Ø
                G=E(JP2)                                                 BNDRØ69Ø
                E(JP2)=B1*E(JP2)                                         BNDRØ7ØØ
 5ØØ            CONTINUE                                                 BNDRØ71Ø
 7ØØ        CONTINUE                                                     BNDRØ72Ø
            U=1.ØEØ                                                      BNDRØ73Ø
            DO 85Ø J=2, N                                                BNDRØ74Ø
                E(J)=SQRT(D(J))                                          BNDRØ75Ø
                D(J)=D(J)*V(J)                                           BNDRØ76Ø
                E(J-1)=U*E(J)*W(J)                                       BNDRØ77Ø
 85Ø            U=E(J)                                                   BNDRØ78Ø
            D(1)=V(1)                                                    BNDRØ79Ø
            E(N)=Ø.ØEØ                                                   BNDRØ8ØØ
            RETURN                                                       BNDRØ81Ø
            END                                                          BNDRØ82Ø
C
C
            SUBROUTINE STEP1(N,NTYPE,TN1,TN2,M,MTYPE,TM1,TM2,            STP1ØØ1Ø
     1          IBDIM,B,ND,ID,R,A)                                       STP1ØØ2Ø
C           SUBROUTINE STEP1 CARRIES OUT THE INITIAL MARCHING PHASE OF   STP1ØØ3Ø
C       THE GENERALIZED MARCHING ALGORITHM.  THE SYSTEM OF EQUATIONS     STP1ØØ4Ø
C       IS REDUCED TO AN EQUIVALENT SET FOR THE UNKNOWNS ON THE ND       STP1ØØ5Ø
C       SEPARATING LINES.                                                STP1ØØ6Ø
C       VERSION DATE: MARCH 1, 1977.                                     STP1ØØ7Ø
                REAL ID                                                  STP1ØØ8Ø
                DIMENSION TN1(1),TN2(1),TM1(1),TM2(1),A(1),B(1),R(1),ID(1) STP1ØØ9Ø
            NN=ND                                                        STP1Ø1ØØ
            IF(NTYPE.GE.3) NN=ND+1                                       STP1Ø11Ø
            IF(NN.LT.1) RETURN                                           STP1Ø12Ø
            M1=M+1                                                       STP1Ø13Ø
            M2=M1+M                                                      STP1Ø14Ø
            ITYPE=NTYPE-(NTYPE/2)*2                                      STP1Ø15Ø
            ID2=ID(2)                                                    STP1Ø16Ø
            DO 13Ø LNINDX=1,NN                                           STP1Ø17Ø
                LINE=ID(LNINDX+1)                                        STP1Ø18Ø
                LINEA=ID(LNINDX+2)                                       STP1Ø19Ø
                LINEB=ID(LNINDX)                                         STP1Ø2ØØ
                LINEC=LINE                                               STP1Ø21Ø
                IF(LINE.EQ.N) LINEC=Ø                                    STP1Ø22Ø
                ISPANA=LINEA-LINE-1                                      STP1Ø23Ø
                IF(ISPANA.LE.Ø) ISPANA=ID2-1                             STP1Ø24Ø
                ISPANB=LINE-LINEB-1                                      STP1Ø25Ø
C               CHECK FOR POSSIBLE REDUCTION IN COMPUTATION IN THE       STP1Ø26Ø
C               CONSTANT COEFFICIENT CASE.                               STP1Ø27Ø
                IF((ITYPE.EQ.1).OR.(ISPANA.NE.ISPANB)) GO TO 1Ø5         STP1Ø28Ø
                IF((TN1(LINEB+1).EQ.TN1(LINEA-1)).AND.                   STP1Ø29Ø
     1              (TN2(LINEB+2).EQ.TN2(LINEA-1))) GO TO 12Ø            STP1Ø3ØØ
C               THE FOLLOWING CODE IS EXECUTED WHEN NO REDUCTION         STP1Ø31Ø
C               IN COMPUTATION IS POSSIBLE.                              STP1Ø32Ø
  1Ø5           CALL MARCH1(M,MTYPE,TM1,TM2,TN1,TN2,IBDIM,B,             STP1Ø33Ø
     1              ISPANB,LINE,-1,A(1),A(M1))                           STP1Ø34Ø
                INDXRT=LINEB+1                                           STP1Ø35Ø
                CALL TRI1(M,MTYPE,TM1,TM2,ISPANB,R(INDXRT),TN2(INDXRT),  STP1Ø36Ø
     1              A(1),A(M1),A(M2))                                    STP1Ø37Ø
                T1=TN2(LINE)                                             STP1Ø38Ø
                LNPTR=(LINE-1)*IBDIM                                     STP1Ø39Ø
                DO 11Ø ICOMP=1,M                                         STP1Ø4ØØ
                    LNPTR=LNPTR+1                                        STP1Ø41Ø
```

```
 11Ø          B(LNPTR)=B(LNPTR)-T1*A(ICOMP)                              STP1Ø42Ø
            CALL MARCH1(M,MTYPE,TM1,TM2,TN1,TN2,IBDIM,B,                STP1Ø43Ø
     1          ISPANA,LINEC,1,A(1),A(M1))                             STP1Ø44Ø
            INDXRT=LINEC+1                                              STP1Ø45Ø
            CALL TRI1(M,MTYPE,TM1,TM2,ISPANA,R(INDXRT),TN2(INDXRT),     STP1Ø46Ø
     1          A(1),A(M1),A(M2))                                       STP1Ø47Ø
            INDXRT=INDXRT+ISPANA                                        STP1Ø48Ø
            T1=TN2(INDXRT)                                              STP1Ø49Ø
            LNPTR=(LINE-1)*IBDIM                                        STP1Ø5ØØ
            DO 115 ICOMP=1,M                                            STP1Ø51Ø
                LNPTR=LNPTR+1                                           STP1Ø52Ø
 115            B(LNPTR)=B(LNPTR)-T1*A(ICOMP)                           STP1Ø53Ø
            GO TO 13Ø                                                   STP1Ø54Ø
C           THE FOLLOWING CODE IS EXECUTED FOR CONSTANT COEFFICIENT     STP1Ø55Ø
C           PROBLEMS IN WHICH A REDUCTION IN COMPUTATION IS POSSIBLE.   STP1Ø56Ø
 12Ø        CALL MARCH3(M,MTYPE,TM1,TM2,TN1,TN2,IBDIM,B,                STP1Ø57Ø
     1          ISPANB,LINE.LINEC,A(1),A(M1))                          STP1Ø58Ø
           INDXRT=LINEB+1                                               STP1Ø59Ø
            CALL TRI1(M,MTYPE,TM1,TM2,ISPANB,R(INDXRT),TN2(INDXRT),     STP1Ø6ØØ
     1          A(1),A(M1),A(M2))                                       STP1Ø61Ø
            T1=TN2(LINE)                                                STP1Ø62Ø
            LNPTR=(LINE-1)*IBDIM                                        STP1Ø63Ø
            DO 125 ICOMP=1,M                                            STP1Ø64Ø
                LNPTR=LNPTR+1                                           STP1Ø65Ø
 125            B(LNPTR)=B(LNPTR)-T1*A(ICOMP)                           STP1Ø66Ø
 13Ø        CONTINUE                                                    STP1Ø67Ø
       RETURN                                                           STP1Ø68Ø
       END                                                              STP1Ø69Ø
C
C
       SUBROUTINE STEP2(N,NTYPE,TN2,M,MTYPE,TM1,TM2,                    STP2ØØ1Ø
     1     IBDIM,B,NE,ND,ID,R,A)                                        STP2ØØ2Ø
C           SUBROUTINE STEP2 USES NE-1 2-REDUCTION STEPS TO REDUCE      STP2ØØ3Ø
C       THE SET OF EQUATIONS FOR THE ND SEPARATING LINES TO A MATRIX    STP2ØØ4Ø
C       EQUATION FOR A SINGLE LINE OF UNKNOWNS.  APPROXIMATELY          STP2ØØ5Ø
C       HALF OF THE REMAINING LINES ARE ELIMINATED AT EACH RECURSIVE    STP2ØØ6Ø
C       STEP.                                                           STP2ØØ7Ø
C       VERSION DATE: MARCH 1, 1977.                                    STP2ØØ8Ø
           REAL ID                                                      STP2ØØ9Ø
           DIMENSION TN2(1),TM1(1),TM2(1),A(1),B(1),R(1),ID(1)          STP2Ø1ØØ
       NN=NE-1                                                          STP2Ø11Ø
       IF(NTYPE.GE.3) NN=NE                                             STP2Ø12Ø
       IF(NN.LT.1) RETURN                                               STP2Ø13Ø
        l=M+1                                                           STP2Ø14Ø
       M2=M1+M                                                          STP2Ø15Ø
       M3=M2+M                                                          STP2Ø16Ø
       ITYPE=NTYPE-(NTYPE/2)*2                                          STP2Ø17Ø
       DO 24Ø I=1,NN                                                    STP2Ø18Ø
           I2=2**I                                                      STP2Ø19Ø
           I1=I2/2                                                      STP2Ø2ØØ
           J1=(I-1)*N+1                                                 STP2Ø21Ø
           J2=I*N+1                                                     STP2Ø22Ø
       DO 24Ø LNINDX=I1,ND,I2                                           STP2Ø23Ø
           LINE=ID(LNINDX+1)                                            STP2Ø24Ø
           LINEA=LNINDX+1+I1                                            STP2Ø25Ø
           LINEA=ID(LINEA)                                              STP2Ø26Ø
           LINEB=LNINDX+1-I1                                            STP2Ø27Ø
           LINEB=ID(LINEB)                                              STP2Ø28Ø
           ISPANA=LINEA-LINE                                            STP2Ø29Ø
           ISPANB=LINE-LINEB                                            STP2Ø3ØØ
           IF((LINEB.EQ.Ø).AND.(NTYPE.LT.3)) GO TO 215                  STP2Ø31Ø
           LNPTR=(LINE-1)*IBDIM                                         STP2Ø32Ø
           DO 2Ø5 ICOMP=1,M                                             STP2Ø33Ø
               LNPTR=LNPTR+1                                            STP2Ø34Ø
 2Ø5           A(ICOMP)=B(LNPTR)                                        STP2Ø35Ø
           INDXRT=LINEB+J2                                              STP2Ø36Ø
           INDXTN=LINEB+1                                               STP2Ø37Ø
           CALL TRI1(M,MTYPE,TM1,TM2,ISPANB,R(INDXRT),TN2(INDXTN),      STP2Ø38Ø
     1          A(1),A(M1),A(M2))                                       STP2Ø39Ø
           IRTDIF=LINE+J1                                               STP2Ø4ØØ
           INDXRT=INDXRT+ISPANB                                         STP2Ø41Ø
           ISPAN=ISPANA-1                                               STP2Ø42Ø
           CALL TRI2(M,MTYPE,TM1,TM2,ISPAN,R(INDXRT),R(IRTDIF),         STP2Ø43Ø
     1          A(1),A(M1),A(M2),A(M3))                                 STP2Ø44Ø
           ISPAN=LINEB                                                  STP2Ø45Ø
           IF(LINEB.EQ.Ø) ISPAN=N                                       STP2Ø46Ø
```

```
            LNPTR=(ISPAN-1)*IBDIM                                          STP2Ø47Ø
            DO 21Ø ICOMP=1,M                                               STP2Ø48Ø
                LNPTR=LNPTR+1                                              STP2Ø49Ø
  21Ø           B(LNPTR)=B(LNPTR)+A(ICOMP)                                 STP2Ø5ØØ
C           CHECK FOR POSSIBLE REDUCTION IN COMPUTATION IN THE             STP2Ø51Ø
C           CONSTANT COEFFICIENT CASE.                                     STP2Ø52Ø
            IF(LINEA.GT.N) GO TO 24Ø                                       STP2Ø53Ø
            IF((ITYPE.EQ.Ø).AND.(ISPANA.EQ.ISPANB)) GO TO 225              STP2Ø54Ø
C           THE FOLLOWING CODE IS EXECUTED WHEN NO REDUCTION               STP2Ø55Ø
C           IN COMPUTATION IS POSSIBLE.                                    STP2Ø56Ø
  215       LNPTR=(LINE-1)*IBDIM                                           STP2Ø57Ø
            DO 22Ø ICOMP=1,M                                               STP2Ø58Ø
                LNPTR=LNPTR+1                                              STP2Ø59Ø
  22Ø           A(ICOMP)=B(LNPTR)                                          STP2Ø6ØØ
            INDXRT=LINEB+J2                                                STP2Ø61Ø
            INDXTN=LINE+1                                                  STP2Ø62Ø
            CALL TRI1(M,MTYPE,TM1,TM2,ISPANA,R(INDXRT),TN2(INDXTN),        STP2Ø63Ø
     1          A(1),A(M1),A(M2))                                          STP2Ø64Ø
            INDXRT=INDXRT+ISPANA                                           STP2Ø65Ø
            IRTDIF=LINEB+J1                                                STP2Ø66Ø
            ISPAN=ISPANB-1                                                 STP2Ø67Ø
            CALL TRI2(M,MTYPE,TM1,TM2,ISPAN,R(INDXRT),R(IRTDIF),           STP2Ø68Ø
     1          A(1),A(M1),A(M2),A(M3))                                    STP2Ø69Ø
  225       LNPTR=(LINEA-1)*IBDIM                                          STP2Ø7ØØ
            DO 23Ø ICOMP=1,M                                               STP2Ø71Ø
                LNPTR=LNPTR+1                                              STP2Ø72Ø
  23Ø           B(LNPTR)=B(LNPTR)+A(ICOMP)                                 STP2Ø73Ø
  24Ø       CONTINUE                                                       STP2Ø74Ø
        RETURN                                                             STP2Ø75Ø
        END                                                                STP2Ø76Ø

        SUBROUTINE STEP3(N,NTYPE,TN2,M,MTYPE,TM1,TM2,                      STP3ØØ1Ø
     1      IBDIM,B,NE,ND,ID,R,A)                                          STP3ØØ2Ø
C           IN SUBROUTINE STEP3, THE UNKNOWNS ON THE ND SEPARATING         STP3ØØ3Ø
C       LINES ARE DETERMINED. THE LINES ARE DETERMINED IN THE              STP3ØØ4Ø
C       REVERSE ORDER THAT THEY WERE ELIMINATED IN STEP2.                  STP3ØØ5Ø
C       VERSION DATE: MARCH 1, 1977.                                       STP3ØØ6Ø
            REAL ID                                                        STP3ØØ7Ø
            DIMENSION TN2(1),TM1(1),TM2(1),A(1),B(1),R(1),ID(1)            STP3ØØ8Ø
        NP1=N+1                                                            STP3ØØ9Ø
        M1=M+1                                                             STP3Ø1ØØ
        M2=M1+M                                                            STP3Ø11Ø
        M3=M2+M                                                            STP3Ø12Ø
        ITYPE=NTYPE-(NTYPE/2)*2                                            STP3Ø13Ø
C       IN THE FOLLOWING SEGMENT OF CODE, THE SPECIAL LINE OF UNKNOWNS     STP3Ø14Ø
C       ASSOCIATED WITH PERIODIC BOUNDARY CONDITIONS IS DETERMINED.        STP3Ø15Ø
  25Ø   IF(NTYPE.LT.3) GO TO 27Ø                                           STP3Ø16Ø
        NM1=N-1                                                            STP3Ø17Ø
        LNPTR=NM1*IBDIM                                                    STP3Ø18Ø
        DO 255 ICOMP=1,M                                                   STP3Ø19Ø
            LNPTR=LNPTR+1                                                  STP3Ø2ØØ
  255       A(ICOMP)=B(LNPTR)                                            STP3Ø21Ø
        INDXRT=(NE+1)*N+1                                                  STP3Ø22Ø
        IRTDIF=NE*N+1                                                      STP3Ø23Ø
        CALL TRI2(M,MTYPE,TM1,TM2,NM1,R(INDXRT),R(IRTDIF),                 STP3Ø24Ø
     1      A(1),A(M1),A(M2),A(M3))                                        STP3Ø25Ø
        INDXRT=INDXRT+NM1                                                  STP3Ø26Ø
        CALL TRI1(M,MTYPE,TM1,TM2,1,R(INDXRT),TN2(NP1),                    STP3Ø27Ø
     1      A(1),A(M1),A(M2))                                              STP3Ø28Ø
        LNPTR=NM1*IBDIM                                                    STP3Ø29Ø
        L2=LNPTR-IBDIM                                                     STP3Ø3ØØ
        T1=TN2(1)                                                          STP3Ø31Ø
        T2=TN2(N)                                                          STP3Ø32Ø
        DO 265 ICOMP=1,M                                                   STP3Ø33Ø
            LNPTR=LNPTR+1                                                  STP3Ø34Ø
            L2=L2+1                                                        STP3Ø35Ø
            X=A(ICOMP)                                                     STP3Ø36Ø
            B(LNPTR)=X                                                     STP3Ø37Ø
            B(L2)=B(L2)+T2*X                                               STP3Ø38Ø
  265       B(ICOMP)=B(ICOMP)+T1*X                                       STP3Ø39Ø
  27Ø   IF(NE.EQ.Ø) RETURN                                                 STP3Ø4ØØ
C       IN THE FOLLOWING SEGMENT OF CODE, THE UNKNOWNS ON THE              STP3Ø41Ø
C       REMAINING SEPARATING LINES ARE DETERMINED.                         STP3Ø42Ø
        DO 345 II=1,NE                                                     STP3Ø43Ø
            I=NE+1-II                                                      STP3Ø44Ø
```

```
                I2=2**I                                                 STP3Ø45Ø
                I1=I2/2                                                 STP3Ø46Ø
                J1=(I-1)*N+1                                            STP3Ø47Ø
                J2=I*N+1                                                STP3Ø48Ø
            DO 345 LNINDX=I1,ND,I2                                      STP3Ø49Ø
                LINE=ID(LNINDX+1)                                       STP3Ø5ØØ
                LINEA=LNINDX+1+I1                                       STP3Ø51Ø
                LINEA=ID(LINEA)                                         STP3Ø52Ø
                LINEB=LNINDX+1-I1                                       STP3Ø53Ø
                LINEB=ID(LINEB)                                         STP3Ø54Ø
                ISPANA=LINEA-LINE-1                                     STP3Ø55Ø
                ISPANB=LINE-LINEB-1                                     STP3Ø56Ø
C           IN THE FOLLOWING SEGMENT OF CODE, THE RIGHT HAND SIDE IS    STP3Ø57Ø
C           MODIFIED USING UNKNOWNS WHICH WERE SOLVED FOR ON A PREVIOUS STP3Ø58Ø
C           STEP OF THE RECURSION.                                      STP3Ø59Ø
                ISPAN=LINEB                                             STP3Ø6ØØ
                IF(LINEB.EQ.Ø) ISPAN=ID(ND+2)                           STP3Ø61Ø
                IF(ISPAN.GT.N) GO TO 295                                STP3Ø62Ø
                LNPTR=(ISPAN-1)*IBDIM                                   STP3Ø63Ø
                DO 275 ICOMP=1,M                                        STP3Ø64Ø
                    LNPTR=LNPTR+1                                       STP3Ø65Ø
  275               A(ICOMP)=B(LNPTR)                                   STP3Ø66Ø
C               CHECK FOR POSSIBLE REDUCTION IN COMPUTATION IN THE      STP3Ø67Ø
C               CONSTANT COEFFICIENT CASE.                              STP3Ø68Ø
                IF((ITYPE.EQ.1).OR.(LINEA.GT.N)) GO TO 285              STP3Ø69Ø
                IF(ISPANA.NE.ISPANB) GO TO 285                          STP3Ø7ØØ
                LNPTR=(LINEA-1)*IBDIM                                   STP3Ø71Ø
                DO 28Ø ICOMP=1,M                                        STP3Ø72Ø
                    LNPTR=LNPTR+1                                       STP3Ø73Ø
  28Ø               A(ICOMP)=A(ICOMP)+B(LNPTR)                          STP3Ø74Ø
                GO TO 3Ø5                                               STP3Ø75Ø
C               THE FOLLOWING CODE IS EXECUTED WHEN NO REDUCTION        STP3Ø76Ø
C               IN COMPUTATION IS POSSIBLE.                             STP3Ø77Ø
  285           INDXRT=LINEB+J1                                         STP3Ø78Ø
                INDXTN=LINEB+1                                          STP3Ø79Ø
                CALL TRI1(M,MTYPE,TM1,TM2,ISPANB,R(INDXRT),TN2(INDXTN), STP3Ø8Ø0
     1              A(1),A(M1),A(M2))                                   STP3Ø81Ø
                T1=TN2(LINE)                                            STP3Ø82Ø
                LNPTR=(LINE-1)*IBDIM                                    STP3Ø83Ø
                DO 29Ø ICOMP=1,M                                        STP3Ø84Ø
                    LNPTR=LNPTR+1                                       STP3Ø85Ø
  29Ø               B(LNPTR)=B(LNPTR)+T1*A(ICOMP)                       STP3Ø86Ø
  295           IF(LINEA.GT.N) GO TO 315                                STP3Ø87Ø
                LNPTR=(LINEA-1)*IBDIM                                   STP3Ø88Ø
                DO 3ØØ ICOMP=1,M                                        STP3Ø89Ø
                    LNPTR=LNPTR+1                                       STP3Ø9ØØ
  3ØØ               A(ICOMP)=B(LNPTR)                                   STP3Ø91Ø
  3Ø5           INDXRT=LINE+J1                                          STP3Ø92Ø
                INDXTN=LINE+1                                           STP3Ø93Ø
                CALL TRI1(M,MTYPE,TM1,TM2,ISPANA,R(INDXRT),TN2(INDXTN), STP3Ø94Ø
     1              A(1),A(M1),A(M2))                                   STP3Ø95Ø
                T1=TN2(LINEA)                                           STP3Ø96Ø
                LNPTR=(LINE-1)*IBDIM                                    STP3Ø97Ø
                DO 31Ø ICOMP=1,M                                        STP3Ø98Ø
                    LNPTR=LNPTR+1                                       STP3Ø99Ø
  31Ø               B(LNPTR)=B(LNPTR)+T1*A(ICOMP)                       STP31ØØØ
C           IN THE FOLLOWING SEGMENT OF CODE, WE SOLVE FOR THE NEW LINE STP31Ø1Ø
C           OF UNKNOWNS, AND THEN MODIFY THE RIGHT HAND SIDES OF THE    STP31Ø2Ø
C           ND+1 SMALLER BLOCKS IN THE APPROPRIATE FASHION.             STP31Ø3Ø
  315           LNPTR=(LINE-1)*IBDIM                                    STP31Ø4Ø
                DO 33Ø ICOMP=1,M                                        STP31Ø5Ø
                    LNPTR=LNPTR+1                                       STP31Ø6Ø
  33Ø               A(ICOMP)=B(LNPTR)                                   STP31Ø7Ø
                INDXRT=LINEB+J2                                         STP31Ø8Ø
                IRTDIF=LINEB+J1                                         STP31Ø9Ø
                CALL TRI2(M,MTYPE,TM1,TM2,ISPANB,R(INDXRT),R(IRTDIF),   STP311ØØ
     1              A(1),A(M1),A(M2),A(M3))                             STP31111Ø
                INDXRT=LINE+J2                                          STP31112Ø
                IRTDIF=LINE+J1                                          STP31113Ø
                CALL TRI1(M,MTYPE,TM1,TM2,1,R(INDXRT-1),TN2(NP1),       STP31114Ø
     1              A(1),A(M1),A(M2))                                   STP31115Ø
                CALL TRI2(M,MTYPE,TM1,TM2,ISPANA,R(INDXRT),R(IRTDIF),   STP31116Ø
     1              A(1),A(M1),A(M2),A(M3))                             STP31117Ø
                LNPTR=(LINE-1)*IBDIM                                    STP31118Ø
                L2=LNPTR-IBDIM                                          STP31119Ø
                L1=LNPTR+IBDIM                                          STP312ØØ
```

```
            T1=TN2(LINE+1)                                                  STP3121Ø
            T2=TN2(LINE)                                                    STP3122Ø
            DO 345 ICOMP=1,M                                                STP3123Ø
                LNPTR=LNPTR+1                                               STP3124Ø
                L1=L1+1                                                     STP3125Ø
                L2=L2+1                                                     STP3126Ø
                X=A(ICOMP)                                                  STP3127Ø
                B(LNPTR)=X                                                  STP3128Ø
                B(L1)=B(L1)+T1*X                                            STP3129Ø
  345           B(L2)=B(L2)+T2*X                                            STP3130Ø
        RETURN                                                              STP3131Ø
        END                                                                 STP3132Ø
C
C
        SUBROUTINE STEP4(TN1,TN2,M,MTYPE,TM1,TM2,IBDIM,B,ND,ID,R,A)         STP4ØØ1Ø
C           IN SUBROUTINE STEP4, THE ND+1 SMALLER PROBLEMS ARE              STP4ØØ2Ø
C       SOLVED USING THE MARCHING ALGORITHM. THIS COMPLETES                 STP4ØØ3Ø
C       THE BACKSOLUTION PHASE OF THE GENERALIZED MARCHING ALGORITHM.       STP4ØØ4Ø
C       VERSION DATE: MARCH 1, 1977.                                        STP4ØØ5Ø
            REAL ID                                                         STP4ØØ6Ø
            DIMENSION TN1(1),TN2(1),TM1(1),TM2(1),A(1),B(1),R(1),ID(1)      STP4ØØ7Ø
        M1=M+1                                                              STP4ØØ8Ø
        M2=M1+M                                                             STP4ØØ9Ø
        NDP1=ND+1                                                           STP4Ø1ØØ
        DO 44Ø LNINDX=1,NDP1                                                STP4Ø11Ø
            LINE=ID(LNINDX)                                                 STP4Ø12Ø
            ISPAN=ID(LNINDX+1)                                              STP4Ø13Ø
            ISPAN=ISPAN-LINE-1                                              STP4Ø14Ø
            CALL MARCH1(M,MTYPE,TM1,TM2,TN1,TN2,IBDIM,B,                    STP4Ø15Ø
     1          ISPAN,LINE,1,A(1),A(M1))                                    STP4Ø16Ø
            INDXRT=LINE+1                                                   STP4Ø17Ø
            INDXTN=INDXRT+1                                                 STP4Ø18Ø
            CALL TRI1(M,MTYPE,TM1,TM2,ISPAN,R(INDXRT),TN2(INDXTN),          STP4Ø19Ø
     1          A(1),A(M1),A(M2))                                           STP4Ø2ØØ
            CALL MARCH2(M,MTYPE,TM1,TM2,TN1,TN2,IBDIM,B,                    STP4Ø21Ø
     1          ISPAN,LINE,A(1),A(M1))                                      STP4Ø22Ø
   44Ø      CONTINUE                                                        STP4Ø23Ø
        RETURN                                                              STP4Ø24Ø
        END                                                                 STP4Ø25Ø
C
C
        SUBROUTINE TRI1(M,MTYPE,TM1,TM2,ISPAN,R,S,V,U,E)                    TRI1ØØ1Ø
C           SUBROUTINE TRI1 SOLVES A LINEAR SYSTEM WHICH INVOLVES           TRI1ØØ2Ø
C       A POLYNOMIAL OF DEGREE ISPAN IN THE MATRIX TM.  THE ZEROES OF       TRI1ØØ3Ø
C       THE POLYNOMIAL ARE A SUBSET OF THE SET OF EIGENVALUES               TRI1ØØ4Ø
C       COMPUTED DURING THE PREPROCESSING PHASE. THE LEADING COEFFICIENT    TRI1ØØ5Ø
C       IS GIVEN AS A PRODUCT OF ISPAN SCALARS STORED IN S.                 TRI1ØØ6Ø
C       THE RIGHT HAND SIDE IS STORED IN V.  THE SOLUTION IS COMPUTED       TRI1ØØ7Ø
C       BY SOLVING ISPAN LINEAR SYSTEMS INVOLVING THE FACTORS OF THE        TRI1ØØ8Ø
C       POLYNOMIAL.  FOUR COMPLETE LOOPS ARE INCLUDED; THE CORRECT LOOP     TRI1ØØ9Ø
C       FOR A GIVEN PROBLEM IS DETERMINED BY THE VALUE OF MTYPE.            TRI1Ø1ØØ
C       VERSION DATE: FEBRUARY 1, 1977.                                     TRI1Ø11Ø
            DIMENSION TM1(1),TM2(1),U(1),V(1),E(1),R(1),S(1)                TRI1Ø12Ø
        MM1=M-1                                                             TRI1Ø13Ø
        GO TO (3,5Ø,2Ø,75),MTYPE                                            TRI1Ø14Ø
C       THE LOOP FOR MTYPE = 1.                                             TRI1Ø15Ø
    3   DO 15 J=1,ISPAN                                                     TRI1Ø16Ø
            D=R(J)                                                          TRI1Ø17Ø
            DS=S(J)                                                         TRI1Ø18Ø
            U(1)=TM1(1)+D                                                   TRI1Ø19Ø
            DO 5 ICOMP=2,M                                                  TRI1Ø2ØØ
                ICM1=ICOMP-1                                                TRI1Ø21Ø
                Q=TM2(ICOMP)/U(ICM1)                                        TRI1Ø22Ø
                V(ICOMP)=V(ICOMP)+V(ICM1)*Q                                 TRI1Ø23Ø
    5           U(ICOMP)=TM1(ICOMP)+D-Q*TM2(ICOMP)                          TRI1Ø24Ø
            Q=V(M)/U(M)                                                     TRI1Ø25Ø
            V(M)=Q*DS                                                       TRI1Ø26Ø
            DO 1Ø II=1,MM1                                                  TRI1Ø27Ø
                ICOMP=M-II                                                  TRI1Ø28Ø
                Q=(V(ICOMP)+TM2(ICOMP+1)*Q)/U(ICOMP)                        TRI1Ø29Ø
   1Ø           V(ICOMP)=Q*DS                                               TRI1Ø3ØØ
   15       CONTINUE                                                        TRI1Ø31Ø
        RETURN                                                              TRI1Ø32Ø
C       THE LOOP FOR MTYPE = 3.                                             TRI1Ø33Ø
   2Ø   DO 4Ø J=1,ISPAN                                                     TRI1Ø34Ø
            D=R(J)                                                          TRI1Ø35Ø
```

```
                DS=S(J)                                                          TRI1Ø36Ø
                U(1)=TM1(1)+D                                                    TRI1Ø37Ø
                E(1)=TM2(1)                                                      TRI1Ø38Ø
                DO 25 ICOMP=2,MM1                                                TRI1Ø39Ø
                    ICM1=ICOMP-1                                                 TRI1Ø4ØØ
                    Q=TM2(ICOMP)/U(ICM1)                                         TRI1Ø41Ø
                    V(ICOMP)=V(ICOMP)+V(ICM1)*Q                                  TRI1Ø42Ø
                    U(ICOMP)=TM1(ICOMP)+D-Q*TM2(ICOMP)                           TRI1Ø43Ø
   25               E(ICOMP)=Q*E(ICM1)                                           TRI1Ø44Ø
                E(MM1)=(E(MM1)+TM2(M))/U(MM1)                                    TRI1Ø45Ø
                V(MM1)=V(MM1)/U(MM1)                                             TRI1Ø46Ø
                DO 3Ø II=2,MM1                                                   TRI1Ø47Ø
                    ICOMP=M-II                                                   TRI1Ø48Ø
                    ICP1=ICOMP+1                                                 TRI1Ø49Ø
                    V(ICOMP)=(V(ICOMP)+TM2(ICP1)*V(ICP1))/U(ICOMP)               TRI1Ø5ØØ
   3Ø               E(ICOMP)=(E(ICOMP)+TM2(ICP1)*E(ICP1))/U(ICOMP)               TRI1Ø51Ø
                Q=TM1(M)+D-TM2(1)*E(1)-TM2(M)*E(MM1)                             TRI1Ø52Ø
                Q=(V(M)+TM2(1)*V(1)+TM2(M)*V(MM1))/Q                             TRI1Ø53Ø
                V(M)=Q*DS                                                        TRI1Ø54Ø
                DO 35 ICOMP=1,MM1                                                TRI1Ø55Ø
   35               V(ICOMP)=(V(ICOMP)+E(ICOMP)*Q)*DS                            TRI1Ø56Ø
   4Ø           CONTINUE                                                         TRI1Ø57Ø
            RETURN                                                               TRI1Ø58Ø
C           THE LOOP FOR MTYPE = 2.                                              TRI1Ø59Ø
   5Ø       MM2=M-2                                                             TRI1Ø6ØØ
            T3=1.ØEØ/TM2(3)                                                     TRI1Ø61Ø
            T2=TM2(2)*T3                                                        TRI1Ø62Ø
            T4=TM2(M)*T3                                                        TRI1Ø63Ø
            DO 65 J=1,ISPAN                                                     TRI1Ø64Ø
                D=R(J)                                                          TRI1Ø65Ø
                DS=S(J)*T3                                                      TRI1Ø66Ø
                U(1)=TM2(3)/(TM1(1)+D)                                          TRI1Ø67Ø
                T1=(TM1(2)+D)*T3                                                TRI1Ø68Ø
                Q=T2*U(1)                                                       TRI1Ø69Ø
                V(2)=V(2)+V(1)*Q                                                TRI1Ø7ØØ
                U(2)=1.ØEØ/(T1-Q*T2)                                            TRI1Ø71Ø
                DO 55 ICOMP=3,MM1                                               TRI1Ø72Ø
                    ICM1=ICOMP-1                                                TRI1Ø73Ø
                    V(ICOMP)=V(ICOMP)+V(ICM1)*U(ICM1)                           TRI1Ø74Ø
   55               U(ICOMP)=1.ØEØ/(T1-U(ICM1))                                 TRI1Ø75Ø
                Q=T4*U(MM1)                                                     TRI1Ø76Ø
                V(M)=V(M)+V(MM1)*Q                                              TRI1Ø77Ø
                U(M)=(TM1(M)+D)*T3-Q*T4                                         TRI1Ø78Ø
                Q=V(M)/U(M)                                                     TRI1Ø79Ø
                V(M)=Q*DS                                                       TRI1Ø8ØØ
                Q=(V(MM1)+Q*T4)*U(MM1)                                          TRI1Ø81Ø
                V(MM1)=Q*DS                                                     TRI1Ø82Ø
                DO 6Ø II=2,MM2                                                  TRI1Ø83Ø
                    ICOMP=M-II                                                  TRI1Ø84Ø
                    Q=(V(ICOMP)+Q)*U(ICOMP)                                     TRI1Ø85Ø
   6Ø               V(ICOMP)=Q*DS                                               TRI1Ø86Ø
                Q=(V(1)+T2*Q)*U(1)                                              TRI1Ø87Ø
                V(1)=Q*DS                                                       TRI1Ø88Ø
   65           CONTINUE                                                        TRI1Ø89Ø
            RETURN                                                              TRI1Ø9ØØ
C           THE LOOP FOR MTYPE = 4.                                             TRI1Ø91Ø
   75       T2=1.ØEØ/TM2(2)                                                     TRI1Ø92Ø
            M1=MM1/2                                                            TRI1Ø93Ø
            M1P1=M1+1                                                           TRI1Ø94Ø
            M1P2=M1+2                                                           TRI1Ø95Ø
            M1P3=M1+3                                                           TRI1Ø96Ø
            MSW=M-(M/2)*2                                                       TRI1Ø97Ø
            M2=M1-MSW                                                           TRI1Ø98Ø
            RM=1.ØEØ                                                            TRI1Ø99Ø
            RA=1.ØEØ                                                            TRI11ØØØ
            IF(MSW.EQ.1) GO TO 8Ø                                               TRI11Ø1Ø
            RM=2.ØEØ                                                            TRI11Ø2Ø
            RA=Ø.ØEØ                                                            TRI11Ø3Ø
   8Ø       DO 85 ICOMP=1,M1                                                    TRI11Ø4Ø
                JCOMP=M+MSW-ICOMP                                               TRI11Ø5Ø
                Q=(V(JCOMP)-V(ICOMP))/2.ØEØ                                     TRI11Ø6Ø
                V(JCOMP)=(V(JCOMP)+V(ICOMP))/2.ØEØ                              TRI11Ø7Ø
   85           V(ICOMP)=Q                                                      TRI11Ø8Ø
            DO 115 J=1,ISPAN                                                    TRI11Ø9Ø
                D=R(J)                                                          TRI1111Ø
                DS=S(J)*T2                                                      TRI11111Ø
```

```
            T1=(TM1(1)+D)*T2                                                TRI11120
            U(1)=1.ØEØ/(T1+RA)                                              TRI11130
            DO 9Ø ICOMP=2,M1                                                TRI11140
                ICM1=ICOMP-1                                                TRI11150
                V(ICOMP)=V(ICOMP)+V(ICM1)*U(ICM1)                           TRI11160
 9Ø             U(ICOMP)=1.ØEØ/(T1-U(ICM1))                                 TRI11170
            U(M1P1)=1.ØEØ/T1                                                TRI11180
            V(M1P2)=V(M1P2)+V(M1P1)*U(M1P1)                                 TRI11190
            U(M1P2)=1.ØEØ/(T1-2.ØEØ*U(M1P1))                                TRI11200
            DO 95 ICOMP=M1P3,MM1                                            TRI11210
                ICM1=ICOMP-1                                                TRI11220
                V(ICOMP)=V(ICOMP)+V(ICM1)*U(ICM1)                           TRI11230
 95             U(ICOMP)=1.ØEØ/(T1-U(ICM1))                                 TRI11240
            V(M)=V(M)+V(MM1)*U(MM1)*RM                                      TRI11250
            U(M)=T1-RM*U(MM1)-RA                                            TRI11260
            Q=V(M)/U(M)                                                     TRI11270
            V(M)=Q*DS                                                       TRI11280
            DO 1Ø5 II=1,M2                                                  TRI11290
                ICOMP=M-II                                                  TRI11300
                Q=(V(ICOMP)+Q)*U(ICOMP)                                     TRI11310
 1Ø5            V(ICOMP)=Q*DS                                               TRI11320
            Q=(V(M1P1)+2.ØEØ*Q)*U(M1P1)                                     TRI11330
            V(M1P1)=Q*DS                                                    TRI11340
            Q=V(M1)*U(M1)                                                   TRI11350
            V(M1)=Q*DS                                                      TRI11360
            DO 11Ø II=2,M1                                                  TRI11370
                ICOMP=M1P1-II                                               TRI11380
                Q=(V(ICOMP)+Q)*U(ICOMP)                                     TRI11390
 11Ø            V(ICOMP)=Q*DS                                               TRI11400
 115          CONTINUE                                                      TRI11410
        DO 12Ø ICOMP=1,M1                                                   TRI11420
            JCOMP=M+MSW-ICOMP                                               TRI11430
            Q=V(JCOMP)-V(ICOMP)                                             TRI11440
            V(JCOMP)=V(JCOMP)+V(ICOMP)                                      TRI11450
 12Ø        V(ICOMP)=Q                                                      TRI11460
        RETURN                                                              TRI11470
        END                                                                 TRI11480


        SUBROUTINE TRI2(M,MTYPE,TM1,TM2,ISPAN,R1,R2,V,W,U,E)                TRI2ØØ1Ø
C           SUBROUTINE TRI2 SOLVES A LINEAR SYSTEM WHICH INVOLVES           TRI2ØØ2Ø
C       A RATIONAL FUNCTION OF THE MATRIX TM.  THE DEGREE OF BOTH THE       TRI2ØØ3Ø
C       NUMERATOR AND THE DENOMINATOR IS ISPAN. THE ZEROES OF BOTH          TRI2ØØ4Ø
C       POLYNOMIALS ARE SUBSETS OF THE EIGENVALUES CALCULATED DURING        TRI2ØØ5Ø
C       THE PREPROCESSING PHASE. THE ZEROES OF THE NUMERATOR ARE            TRI2ØØ6Ø
C       STORED IN R1; THE ZEROES OF THE DENOMINATOR ARE STORED IN R2.       TRI2ØØ7Ø
C       THE LEADING COEFFICIENT OF BOTH POLYNOMIALS IS 1.Ø.  THE            TRI2ØØ8Ø
C       RIGHT HAND SIDE IS STORED IN V.  THE SOLUTION IS COMPUTED BY        TRI2ØØ9Ø
C       SOLVING ISPAN LINEAR SYSTEMS INVOLVING THE FACTORS OF THE           TRI2Ø1ØØ
C       POLYNOMIAL WHICH APPEARS IN THE NUMERATOR.  A CHECK FOR             TRI2Ø11Ø
C       COMMON FACTORS IS CARRIED OUT, AND THEY ARE CANCELLED IF            TRI2Ø12Ø
C       FOUND.    FOUR COMPLETE LOOPS ARE PROVIDED; THE CORECT LOOP         TRI2Ø13Ø
C       FOR A GIVEN PROBLEM IS DETERMINED BY THE VALUE OF MTYPE.            TRI2Ø14Ø
C       VERSION DATE: FEBRUARY 1, 1977.                                     TRI2Ø15Ø
            DIMENSION TM1(1),TM2(1),V(1),U(1),W(1),E(1),R1(1),R2(1)         TRI2Ø16Ø
            COMMON /MACHEP/TOL                                              TRI2Ø17Ø
        TOL2=8.ØEØ*TOL                                                      TRI2Ø18Ø
        MM1=M-1                                                             TRI2Ø19Ø
        GO TO (3,5Ø,2Ø,75),MTYPE                                            TRI2Ø2ØØ
C       THE LOOP FOR MTYPE = 1.                                             TRI2Ø21Ø
    3   DO 15 J=1,ISPAN                                                     TRI2Ø22Ø
            D=R1(J)                                                         TRI2Ø23Ø
            DR=R2(J)-D                                                      TRI2Ø24Ø
            U(1)=TM1(1)+D                                                   TRI2Ø25Ø
            Q=ABS(DR/U(1))                                                  TRI2Ø26Ø
            IF(Q.LE.TOL2) GO TO 15                                          TRI2Ø27Ø
            W(1)=V(1)                                                       TRI2Ø28Ø
            DO 5 ICOMP=2,M                                                  TRI2Ø29Ø
                ICM1=ICOMP-1                                                TRI2Ø3ØØ
                Q=TM2(ICOMP)/U(ICM1)                                        TRI2Ø31Ø
                W(ICOMP)=V(ICOMP)+W(ICM1)*Q                                 TRI2Ø32Ø
    5           U(ICOMP)=TM1(ICOMP)+D-Q*TM2(ICOMP)                          TRI2Ø33Ø
            Q=W(M)/U(M)                                                     TRI2Ø34Ø
            V(M)=V(M)+DR*Q                                                  TRI2Ø35Ø
            DO 1Ø II=1,MM1                                                  TRI2Ø36Ø
                ICOMP=M-II                                                  TRI2Ø37Ø
```

```
                Q=(W(ICOMP)+TM2(ICOMP+1)*Q)/U(ICOMP)                    TRI2Ø38Ø
   1Ø           V(ICOMP)=V(ICOMP)+DR*Q                                  TRI2Ø39Ø
   15       CONTINUE                                                    TRI2Ø4ØØ
        RETURN                                                          TRI2Ø41Ø
C       THE LOOP FOR MTYPE = 3.                                         TRI2Ø42Ø
   2Ø   DO 4Ø J=1,ISPAN                                                 TRI2Ø43Ø
            D=R1(J)                                                     TRI2Ø44Ø
            DR=R2(J)-D                                                  TRI2Ø45Ø
            U(1)=TM1(1)+D                                               TRI2Ø46Ø
            Q=ABS(DR/U(1))                                              TRI2Ø47Ø
            IF(Q.LE.TOL2) GO TO 4Ø                                      TRI2Ø48Ø
            W(1)=V(1)                                                   TRI2Ø49Ø
            E(1)=TM2(1)                                                 TRI2Ø5ØØ
            DO 25 ICOMP=2,MM1                                           TRI2Ø51Ø
                ICM1=ICOMP-1                                            TRI2Ø52Ø
                Q=TM2(ICOMP)/U(ICM1)                                    TRI2Ø53Ø
                W(ICOMP)=V(ICOMP)+W(ICM1)*Q                             TRI2Ø54Ø
                U(ICOMP)=TM1(ICOMP)+D-Q*TM2(ICOMP)                      TRI2Ø55Ø
   25           E(ICOMP)=Q*E(ICM1)                                      TRI2Ø56Ø
            E(MM1)=(E(MM1)+TM2(M))/U(MM1)                               TRI2Ø57Ø
            W(MM1)=W(MM1)/U(MM1)                                        TRI2Ø58Ø
            DO 3Ø II=2,MM1                                              TRI2Ø59Ø
                ICOMP=M-II                                              TRI2Ø6ØØ
                ICP1=ICOMP+1                                            TRI2Ø61Ø
                W(ICOMP)=(W(ICOMP)+TM2(ICP1)*W(ICP1))/U(ICOMP)          TRI2Ø62Ø
   3Ø           E(ICOMP)=(E(ICOMP)+TM2(ICP1)*E(ICP1))/U(ICOMP)          TRI2Ø63Ø
            Q=TM1(M)+D-TM2(1)*E(1)-TM2(M)*E(MM1)                        TRI2Ø64Ø
            Q=(V(M)+TM2(1)*W(1)+TM2(M)*W(MM1))/Q                        TRI2Ø65Ø
            V(M)=V(M)+DR*Q                                              TRI2Ø66Ø
            DO 35 ICOMP=1,MM1                                           TRI2Ø67Ø
   35           V(ICOMP)=V(ICOMP)+DR*(W(ICOMP)+E(ICOMP)*Q)              TRI2Ø68Ø
   4Ø       CONTINUE                                                    TRI2Ø69Ø
        RETURN                                                          TRI2Ø7ØØ
C       THE LOOP FOR MTYPE = 2.                                         TRI2Ø71Ø
   5Ø   MM2=M-2                                                         TRI2Ø72Ø
        T3=1.ØEØ/TM2(3)                                                 TRI2Ø73Ø
        T2=TM2(2)*T3                                                    TRI2Ø74Ø
        T4=TM2(M)*T3                                                    TRI2Ø75Ø
        DO 65 J=1,ISPAN                                                 TRI2Ø76Ø
            D=R1(J)                                                     TRI2Ø77Ø
            DR=(R2(J)-D)*T3                                             TRI2Ø78Ø
            U(1)=TM2(3)/(TM1(1)+D)                                      TRI2Ø79Ø
            Q=ABS(DR*U(1))                                              TRI2Ø8ØØ
            IF(Q.LE.TOL2) GO TO 65                                      TRI2Ø81Ø
            W(1)=V(1)                                                   TRI2Ø82Ø
            T1=(TM1(2)+D)*T3                                            TRI2Ø83Ø
            Q=T2*U(1)                                                   TRI2Ø84Ø
            W(2)=V(2)+W(1)*Q                                            TRI2Ø85Ø
            U(2)=1.ØEØ/(T1-Q*T2)                                        TRI2Ø86Ø
            DO 55 ICOMP=3,MM1                                           TRI2Ø87Ø
                ICM1=ICOMP-1                                            TRI2Ø88Ø
                W(ICOMP)=V(ICOMP)+W(ICM1)*U(ICM1)                       TRI2Ø89Ø
   55           U(ICOMP)=1.ØEØ/(T1-U(ICM1))                             TRI2Ø9ØØ
            Q=T4*U(MM1)                                                 TRI2Ø91Ø
            W(M)=V(M)+W(MM1)*Q                                          TRI2Ø92Ø
            U(M)=(TM1(M)+D)*T3-Q*T4                                     TRI2Ø93Ø
            Q=W(M)/U(M)                                                 TRI2Ø94Ø
            V(M)=V(M)+DR*Q                                              TRI2Ø95Ø
            Q=(W(MM1)+Q*T4)*U(MM1)                                      TRI2Ø96Ø
            V(MM1)=V(MM1)+DR*Q                                          TRI2Ø97Ø
            DO 6Ø II=2,MM2                                              TRI2Ø98Ø
                ICOMP=M-II                                              TRI2Ø99Ø
                Q=(W(ICOMP)+Q)*U(ICOMP)                                 TRI21ØØØ
   6Ø           V(ICOMP)=V(ICOMP)+DR*Q                                  TRI21Ø1Ø
            Q=(W(1)+T2*Q)*U(1)                                          TRI21Ø2Ø
            V(1)=V(1)+DR*Q                                              TRI21Ø3Ø
   65       CONTINUE                                                    TRI21Ø4Ø
        RETURN                                                          TRI21Ø5Ø
C       THE LOOP FOR MTYPE = 4.                                         TRI21Ø6Ø
   75   T2=1.ØEØ/TM2(2)                                                 TRI21Ø7Ø
        M1=MM1/2                                                        TRI21Ø8Ø
        M1P1=M1+1                                                       TRI21Ø9Ø
        M1P2=M1+2                                                       TRI211ØØ
        M1P3=M1+3                                                       TRI2111Ø
        MSW=M-(M/2)*2                                                   TRI2112Ø
        M2=M1-MSW                                                       TRI2113Ø
```

```
      RM=1.ØEØ                                                          TRI2114Ø
      RA=1.ØEØ                                                          TRI2115Ø
      IF(MSW.EQ.1) GO TO 8Ø                                             TRI2116Ø
      RM=2.ØEØ                                                          TRI2117Ø
      RA=Ø.ØEØ                                                          TRI2118Ø
 8Ø   DO 85 ICOMP=1,M1                                                  TRI2119Ø
        JCOMP=M+MSW-ICOMP                                               TRI2120Ø
        Q=(V(JCOMP)-V(ICOMP))/2.ØEØ                                     TRI2121Ø
        V(JCOMP)=(V(JCOMP)+V(ICOMP))/2.ØEØ                              TRI2122Ø
 85     V(ICOMP)=Q                                                      TRI2123Ø
      DO 115 J=1,ISPAN                                                  TRI2124Ø
        D=R1(J)                                                         TRI2125Ø
        DR=(R2(J)-D)*T2                                                 TRI2126Ø
        T1=(TM1(1)+D)*T2                                                TRI2127Ø
        U(1)=1.ØEØ/(T1+RA)                                              TRI2128Ø
        Q=ABS(DR*U(1))                                                  TRI2129Ø
        IF(Q.LE.TOL2) GO TO 115                                         TRI2130Ø
        W(1)=V(1)                                                       TRI2131Ø
        DO 9Ø ICOMP=2,M1                                                TRI2132Ø
          ICM1=ICOMP-1                                                  TRI2133Ø
          W(ICOMP)=V(ICOMP)+W(ICM1)*U(ICM1)                             TRI2134Ø
 9Ø       U(ICOMP)=1.ØEØ/(T1-U(ICM1))                                   TRI2135Ø
        U(M1P1)=1.ØEØ/T1                                                TRI2136Ø
        W(M1P1)=V(M1P1)                                                 TRI2137Ø
        W(M1P2)=V(M1P2)+W(M1P1)*U(M1P1)                                 TRI2138Ø
        U(M1P2)=1.ØEØ/(T1-2.ØEØ*U(M1P1))                                TRI2139Ø
        DO 95 ICOMP=M1P3,MM1                                            TRI2140Ø
          ICM1=ICOMP-1                                                  TRI2141Ø
          W(ICOMP)=V(ICOMP)+W(ICM1)*U(ICM1)                             TRI2142Ø
 95       U(ICOMP)=1.ØEØ/(T1-U(ICM1))                                   TRI2143Ø
        W(M)=V(M)+W(MM1)*U(MM1)*RM                                      TRI2144Ø
        U(M)=T1-RM*U(MM1)-RA                                            TRI2145Ø
        Q=W(M)/U(M)                                                     TRI2146Ø
        V(M)=V(M)+DR*Q                                                  TRI2147Ø
        DO 1Ø5 II=1,M2                                                  TRI2148Ø
          ICOMP=M-II                                                    TRI2149Ø
          Q=(W(ICOMP)+Q)*U(ICOMP)                                       TRI2150Ø
1Ø5       V(ICOMP)=V(ICOMP)+DR*Q                                        TRI2151Ø
        Q=(W(M1P1)+2.ØEØ*Q)*U(M1P1)                                     TRI2152Ø
        V(M1P1)=V(M1P1)+DR*Q                                            TRI2153Ø
        Q=W(M1)*U(M1)                                                   TRI2154Ø
        V(M1)=V(M1)+DR*Q                                                TRI2155Ø
        DO 11Ø II=2,M1                                                  TRI2156Ø
          ICOMP=M1P1-II                                                 TRI2157Ø
          Q=(W(ICOMP)+Q)*U(ICOMP)                                       TRI2158Ø
11Ø       V(ICOMP)=V(ICOMP)+DR*Q                                        TRI2159Ø
115       CONTINUE                                                      TRI2160Ø
      DO 12Ø ICOMP=1,M1                                                 TRI2161Ø
        JCOMP=M+MSW-ICOMP                                               TRI2162Ø
        Q=V(JCOMP)-V(ICOMP)                                             TRI2163Ø
        V(JCOMP)=V(JCOMP)+V(ICOMP)                                      TRI2164Ø
12Ø     V(ICOMP)=Q                                                      TRI2165Ø
      RETURN                                                            TRI2166Ø
      END                                                               TRI2167Ø
C
C
      SUBROUTINE MARCH1(M,MTYPE,TM1,TM2,TN1,TN2,IBDIM,B,                MCH1ØØ1Ø
     1    ISPAN,LINEØ,IALPHA,V1,V2)                                     MCH1ØØ2Ø
C       IN SUBROUTINE MARCH1, THE INITIAL MARCHING PHASE OF THE         MCH1ØØ3Ø
C     MARCHING ALGORITHM IS CARRIED OUT.  ISPAN IS THE NUMBER OF LINES  MCH1ØØ4Ø
C     PRESENT. INFORMATION ABOUT THE DIRECTION OF THE MARCH, AND        MCH1ØØ5Ø
C     THE SCALARS WHICH MULTIPLY THE MARCHING VECTORS AT EACH STAGE     MCH1ØØ6Ø
C     IS STORED IN IALPHA.  THE BOUNDARY LINE FOR THE MARCH IS LINEØ.   MCH1ØØ7Ø
C     THE MARCHING VECTORS ARE V1 AND V2.  THE RESULT IS RETURNED IN    MCH1ØØ8Ø
C     THE VECTOR V1.                                                    MCH1ØØ9Ø
C     VERSION DATE: MARCH 1, 1977.                                      MCH1Ø1ØØ
        DIMENSION TM1(1),TM2(1),TN1(1),TN2(1),B(1),V1(1),V2(1)          MCH1Ø11Ø
      LINE=LINEØ+IALPHA                                                 MCH1Ø12Ø
      IBETA=(IALPHA+1)/2                                                MCH1Ø13Ø
      IBETA=LINE+IBETA                                                  MCH1Ø14Ø
      BETA=TN2(IBETA)                                                   MCH1Ø15Ø
      BETA1=-1.ØEØ/BETA                                                 MCH1Ø16Ø
      LNPTR=(LINE-1)*IBDIM                                              MCH1Ø17Ø
      DO 5 ICOMP=1,M                                                    MCH1Ø18Ø
        LNPTR=LNPTR+1                                                   MCH1Ø19Ø
        V1(ICOMP)=B(LNPTR)*BETA1                                        MCH1Ø2ØØ
```

```
    5       V2(ICOMP)=Ø.ØEØ                                          MCH1Ø21Ø
            IF(ISPAN.EQ.1) RETURN                                    MCH1Ø22Ø
            MM1=M-1                                                  MCH1Ø23Ø
            TC=Ø.ØEØ                                                 MCH1Ø24Ø
            IF(MTYPE.GE.3) TC=TM2(1)                                 MCH1Ø25Ø
            DO 2Ø I=2,ISPAN                                          MCH1Ø26Ø
                LINE=LINE+IALPHA                                     MCH1Ø27Ø
                IBETA=IBETA+IALPHA                                   MCH1Ø28Ø
                LNPTR=(LINE-1)*IBDIM                                 MCH1Ø29Ø
                ALPHA=TN1(LINE)                                      MCH1Ø3ØØ
                BETA2=BETA                                           MCH1Ø31Ø
                BETA=TN2(IBETA)                                      MCH1Ø32Ø
                BETA1=1.ØEØ/BETA                                     MCH1Ø33Ø
                XM=V1(M)                                             MCH1Ø34Ø
                XB=V1(1)                                             MCH1Ø35Ø
                XC=XB                                                MCH1Ø36Ø
                TB=TC                                                MCH1Ø37Ø
                DO 15 ICOMP=1,MM1                                    MCH1Ø38Ø
                    LNPTR=LNPTR+1                                    MCH1Ø39Ø
                    XT=XM                                            MCH1Ø4ØØ
                    XM=XB                                            MCH1Ø41Ø
                    XB=V1(ICOMP+1)                                   MCH1Ø42Ø
                    TT=TB                                            MCH1Ø43Ø
                    TB=TM2(ICOMP+1)                                  MCH1Ø44Ø
                    Q=B(LNPTR)+BETA2*V2(ICOMP)+TT*XT+TB*XB           MCH1Ø45Ø
                    V1(ICOMP)=((TM1(ICOMP)+ALPHA)*XM-Q)*BETA1        MCH1Ø46Ø
   15               V2(ICOMP)=XM                                     MCH1Ø47Ø
                Q=B(LNPTR+1)+BETA2*V2(M)+TB*XM+TC*XC                 MCH1Ø48Ø
                V1(M)=((TM1(M)+ALPHA)*XB-Q)*BETA1                    MCH1Ø49Ø
   2Ø           V2(M)=XB                                             MCH1Ø5ØØ
            RETURN                                                   MCH1Ø51Ø
            END                                                      MCH1Ø52Ø
C
C
            SUBROUTINE MARCH2(M,MTYPE,TM1,TM2,TN1,TN2,IBDIM,B,       MCH2ØØ1Ø
     1          ISPAN,LINEØ,V1,V2)                                   MCH2ØØ2Ø
C               IN SUBROUTINE MARCH2, THE FINAL MARCHING PHASE OF THE MCH2ØØ3Ø
C           MARCHING ALGORITHM IS CARRIED OUT.  ISPAN IS THE NUMBER OF LINESMCH2ØØ4Ø
C           PRESENT.  THE BOUNDARY LINE FOR THE MARCH IS LINEØ. THE MARCHINGMCH2ØØ5Ø
C           VECTORS ARE V1 AND V2. AS THE MARCHING PROCEEDS, THE SOLUTION ISMCH2ØØ6Ø
C           WRITTEN OVER THE RIGHT HAND SIDE B.                      MCH2ØØ7Ø
C           VERSION DATE: MARCH 1, 1977.                             MCH2ØØ8Ø
                DIMENSION TM1(1),TM2(1),TN1(1),TN2(1),B(1),V1(1),V2(1) MCH2ØØ9Ø
            LINE=LINEØ                                               MCH2Ø1ØØ
            DO 5 ICOMP=1,M                                           MCH2Ø11Ø
                V1(ICOMP)=-V1(ICOMP)                                 MCH2Ø12Ø
    5       V2(ICOMP)=Ø.ØEØ                                          MCH2Ø13Ø
            IF(ISPAN.EQ.1) GO TO 25                                  MCH2Ø14Ø
            MM1=M-1                                                  MCH2Ø15Ø
            BETA=TN2(LINE+1)                                         MCH2Ø16Ø
            TC=Ø.ØEØ                                                 MCH2Ø17Ø
            IF(MTYPE.GE.3) TC=TM2(1)                                 MCH2Ø18Ø
            DO 2Ø I=2,ISPAN                                          MCH2Ø19Ø
                LINE=LINE+1                                          MCH2Ø2ØØ
                LNPTR=(LINE-1)*IBDIM                                 MCH2Ø21Ø
                ALPHA=TN1(LINE)                                      MCH2Ø22Ø
                BETA2=BETA                                           MCH2Ø23Ø
                BETA=TN2(LINE+1)                                     MCH2Ø24Ø
                BETA1=1.ØEØ/BETA                                     MCH2Ø25Ø
                XM=V1(M)                                             MCH2Ø26Ø
                XB=V1(1)                                             MCH2Ø27Ø
                XC=XB                                                MCH2Ø28Ø
                TB=TC                                                MCH2Ø29Ø
                DO 15 ICOMP=1,MM1                                    MCH2Ø3ØØ
                    LNPTR=LNPTR+1                                    MCH2Ø31Ø
                    XT=XM                                            MCH2Ø32Ø
                    XM=XB                                            MCH2Ø33Ø
                    XB=V1(ICOMP+1)                                   MCH2Ø34Ø
                    TT=TB                                            MCH2Ø35Ø
                    TB=TM2(ICOMP+1)                                  MCH2Ø36Ø
                    Q=B(LNPTR)+BETA2*V2(ICOMP)+TT*XT+TB*XB           MCH2Ø37Ø
                    V1(ICOMP)=((TM1(ICOMP)+ALPHA)*XM-Q)*BETA1        MCH2Ø38Ø
                    V2(ICOMP)=XM                                     MCH2Ø39Ø
   15               B(LNPTR)=XM                                      MCH2Ø4ØØ
                Q=B(LNPTR+1)+BETA2*V2(M)+TB*XM+TC*XC                 MCH2Ø41Ø
                V1(M)=((TM1(M)+ALPHA)*XB-Q)*BETA1                    MCH2Ø42Ø
```

```
            V2(M)=XB                                                    MCH2Ø43Ø
   2Ø       B(LNPTR+1)=XB                                               MCH2Ø44Ø
   25   LNPTR=LINE*IBDIM                                                MCH2Ø45Ø
        DO 3Ø ICOMP=1,M                                                 MCH2Ø46Ø
            LNPTR=LNPTR+1                                               MCH2Ø47Ø
   3Ø       B(LNPTR)=V1(ICOMP)                                          MCH2Ø48Ø
        RETURN                                                          MCH2Ø49Ø
        END                                                             MCH2Ø5ØØ
C
C
        SUBROUTINE MARCH3(M,MTYPE,TM1,TM2,TN1,TN2,IBDIM,B,              MCH3ØØ1Ø
     1      ISPAN,LINEØ,LINEC,V1,V2)                                    MCH3ØØ2Ø
C           IN SUBROUTINE MARCH3, THE INITIAL MARCHING PAHSE OF THE     MCH3ØØ3Ø
C       MARCHING ALGORITHM IS CARRIED OUT, FOR CONSTANT COEFFICIENT     MCH3ØØ4Ø
C       PROBLEMS MEETING CERTAIN REQUIREMENTS.  ISPAN IS THE NUMBER OF  MCH3ØØ5Ø
C       LINES PRESENT.  MARCHING OCCURS IN BOTH DRRECTIONS FROM THE     MCH3ØØ6Ø
C       BOUNDARY LINE LINEØ.  THE MARCHING VECTORS ARE V1 AND V2.       MCH3ØØ7Ø
C       VERSION DATE: MARCH 1, 1977.                                    MCH3ØØ8Ø
            DIMENSION TM1(1),TM2(1),TN1(1),TN2(1),B(1),V1(1),V2(1)      MCH3ØØ9Ø
        LINE=LINEØ-1                                                    MCH3Ø1ØØ
        LINN=LINEC+1                                                    MCH3Ø11Ø
        BETA=TN2(LINE)                                                  MCH3Ø12Ø
        BETA1=-1.ØEØ/BETA                                               MCH3Ø13Ø
        LNPTR=(LINE-1)*IBDIM                                            MCH3Ø14Ø
        NNPTR=(LINN-1)*IBDIM                                            MCH3Ø15Ø
        DO 5 ICOMP=1,M                                                  MCH3Ø16Ø
            LNPTR=LNPTR+1                                               MCH3Ø17Ø
            NNPTR=NNPTR+1                                               MCH3Ø18Ø
            V1(ICOMP)=(B(LNPTR)+B(NNPTR))*BETA1                         MCH3Ø19Ø
    5       V2(ICOMP)=Ø.ØEØ                                             MCH3Ø2ØØ
        IF(ISPAN.EQ.1) RETURN                                           MCH3Ø21Ø
        MM1=M-1                                                         MCH3Ø22Ø
        TC=Ø.ØEØ                                                        MCH3Ø23Ø
        IF(MTYPE.GE.3) TC=TM2(1)                                        MCH3Ø24Ø
        DO 2Ø I=2,ISPAN                                                 MCH3Ø25Ø
            LINE=LINE-1                                                 MCH3Ø26Ø
            LINN=LINN+1                                                 MCH3Ø27Ø
            LNPTR=(LINE-1)*IBDIM                                        MCH3Ø28Ø
            NNPTR=(LINN-1)*IBDIM                                        MCH3Ø29Ø
            ALPHA=TN1(LINE)                                             MCH3Ø3ØØ
            BETA2=BETA                                                  MCH3Ø31Ø
            BETA=TN2(LINE)                                              MCH3Ø32Ø
            BETA1=1.ØEØ/BETA                                            MCH3Ø33Ø
            XM=V1(M)                                                    MCH3Ø34Ø
            XB=V1(1)                                                    MCH3Ø35Ø
            XC=XB                                                       MCH3Ø36Ø
            TB=TC                                                       MCH3Ø37Ø
            DO 15 ICOMP=1,MM1                                           MCH3Ø38Ø
                LNPTR=LNPTR+1                                           MCH3Ø39Ø
                NNPTR=NNPTR+1                                           MCH3Ø4ØØ
                XT=XM                                                   MCH3Ø41Ø
                XM=XB                                                   MCH3Ø42Ø
                XB=V1(ICOMP+1)                                          MCH3Ø43Ø
                TT=TB                                                   MCH3Ø44Ø
                TB=TM2(ICOMP+1)                                         MCH3Ø45Ø
                Q=B(LNPTR)+B(NNPTR)+BETA2*V2(ICOMP)+TT*XT+TB*XB         MCH3Ø46Ø
                V1(ICOMP)=((TM1(ICOMP)+ALPHA)*XM-Q)*BETA1               MCH3Ø47Ø
   15           V2(ICOMP)=XM                                            MCH3Ø48Ø
            Q=B(LNPTR+1)+B(NNPTR+1)+BETA2*V2(M)+TB*XM+TC*XC             MCH3Ø49Ø
            V1(M)=((TM1(M)+ALPHA)*XB-Q)*BETA1                           MCH3Ø5ØØ
   2Ø       V2(M)=XB                                                    MCH3Ø51Ø
        RETURN                                                          MCH3Ø52Ø
        END                                                             MCH3Ø53Ø
C
C
        SUBROUTINE GMAS(NPC,N,NTYPE,TN1,TN2,M,MTYPE,TM1,TM2,            GMASØØ1Ø
     1      IBDIM,B,VN,VM,K,R,A)                                        GMASØØ2Ø
C           SUBROUTINE GMAS IS USED IN PLACE OF SUBROUTINE GMA FOR      GMASØØ3Ø
C       SOLVING, IN THE LEAST SQUARES SENSE, LINEAR SYSTEMS IN WHICH    GMASØØ4Ø
C       THE MATRIX IS POSITIVE OR NEGATIVE SEMI-DEFINITE HAVING         GMASØØ5Ø
C       ZERO AS AN EIGENVALUE OF MULTIPLICITY ONE.                      GMASØØ6Ø
C           THE LINEAR SYSTEM HAS THE FORM:                             GMASØØ7Ø
C                                                                       GMASØØ8Ø
C       ( TM1(I) + TN1(J) ) * X(I,J)                                    GMASØØ9Ø
C           -  TM2(I+1) * X(I+1,J)  -  TM2(I) * X(I-1,J)                GMASØ1ØØ
C           -  TN2(J+1) * X(I,J+1)  -  TN2(J) * X(I,J-1)  =  B(I,J)     GMASØ11Ø
```

```
C                                                                       GMASØ12Ø
C       FOR I = 1,2,...,M, AND J = 1,2,...,N, WHERE TM2(M+1) IS         GMASØ13Ø
C       INTERPRETED AS TM2(1),  TN2(N+1) AS TN2(1),  X(Ø,J) AS X(M,J),  GMASØ14Ø
C       X(M+1,J) AS X(1,J),  X(I,Ø) AS X(I,N),  AND X(I,N+1) AS X(I,1). GMASØ15Ø
C                                                                       GMASØ16Ø
C       THE PARAMETER LIST:                                             GMASØ17Ø
C                                                                       GMASØ18Ø
C       NPC IS AN INTEGER.  IF NPC = Ø, GMAS DOES NECESSARY             GMASØ19Ø
C           PREPROCESSING CALCULATIONS.  IF NPC = 1, GMAS SOLVES THE    GMASØ2ØØ
C           LINEAR SYSTEM.                                              GMASØ21Ø
C       N IS AN INTEGER, AS DEFINED ABOVE. N MUST BE GREATER THAN 2.    GMASØ22Ø
C       NTYPE IS AN INTEGER DESCRIBING THE ARRAYS TN1 AND TN2.          GMASØ23Ø
C           NTYPE = 1: GENERAL SEPARABLE - DIRICHLET, NEUMANN, OR       GMASØ24Ø
C           MIXED BOUNDARY CONDITIONS.                                  GMASØ25Ø
C           TN1(I) IS ARBITRARY, I = 1,2,...,N.                         GMASØ26Ø
C           TN2(1) = Ø.Ø; TN2(I) IS ARBITRARY, NONZERO, I = 2,3,...,N.  GMASØ27Ø
C           NTYPE = 2: CONSTANT COEFFICIENT - DIRICHLET, NEUMANN,       GMASØ28Ø
C           OR MIXED BOUNDARY CONDITIONS.                               GMASØ29Ø
C           TN1(I) = ALPHA, ALPHA AN ARBITRARY CONSTANT, I = 2,3,...N-1.GMASØ3ØØ
C           TN2(1) = Ø.Ø; TN2(I) = BETA, BETA AN ARBITRARY NONZERO      GMASØ31Ø
C           CONSTANT, I = 3,...,N-1.                                    GMASØ32Ø
C           TOP BOUNDARY CONDITION : ONE OF                             GMASØ33Ø
C           TN1(1) = ALPHA;  TN2(2) = BETA; (DIRICHLET)                 GMASØ34Ø
C           TN1(1) = ALPHA;  TN2(2) = SQRT(2) * BETA; (NEUMANN-CENTERED)GMASØ35Ø
C           TN1(1) = ALPHA - BETA;  TN2(2) = BETA;  (NEUMANN-STAGGERED) GMASØ36Ø
C           TN1(1) = RHO, RHO ARBITRARY; TN2(2) = ZETA, ZETA ARBITRARY, GMASØ37Ø
C           NONZERO; (MIXED).                                           GMASØ38Ø
C           BOTTOM BOUNDARY CONDITION : ONE OF                          GMASØ39Ø
C           TN1(N) = ALPHA;  TN2(N) = BETA; (DIRICHLET)                 GMASØ4ØØ
C           TN1(N) = ALPHA;  TN2(N) = SQRT(2) * BETA; (NEUMANN-CENTERED)GMASØ41Ø
C           TN1(N) = ALPHA - BETA;  TN2(N) = BETA;  (NEUMANN-STAGGERED) GMASØ42Ø
C           TN1(N) = CHI, CHI ARBITRARY; TN2(N) = ETA, ETA ARBITRARY,   GMASØ43Ø
C           NONZERO; (MIXED).                                           GMASØ44Ø
C           NTYPE = 3: GENERAL SEPARABLE - PERIODIC BOUNDARY CONDITIONS.GMASØ45Ø
C           TN1(I) IS ARBITRARY, I = 1,2,...,N.                         GMASØ46Ø
C           TN2(I) IS ARBITRARY, NONZERO, I = 1,2,...,N.                GMASØ47Ø
C           NTYPE = 4: CONSTANT COEFFICIENT - PERIODIC BOUNDARY         GMASØ48Ø
C           CONDITIONS.                                                 GMASØ49Ø
C           TN1(I) = ALPHA, ALPHA ARBITRARY, I = 1,2,...,N.             GMASØ5ØØ
C           TN2(I) = BETA, BETA ARBITRARY, NONZERO, I = 1,2,...,N.      GMASØ51Ø
C           IF N = 3, GMAS MAY RESPECIFY NTYPE.                         GMASØ52Ø
C       TN1 IS AN ARRARY OF LENGTH N+1, DEFINED ABOVE.                  GMASØ53Ø
C       TN2 IS AN ARRAY OF LENGTH N+1, DEFINED ABOVE.                   GMASØ54Ø
C       M IS AN INTEGER, AS DEFINED ABOVE. M MUST BE GREATER THAN 2.    GMASØ55Ø
C       MTYPE IS AN INTEGER DESCRIBING THE ARRAYS TM1 AND TM2.          GMASØ56Ø
C           MTYPE = 1: GENERAL SEPARABLE - DIRICHLET, NEUMANN, OR       GMASØ57Ø
C           MIXED BOUNDARY CONDITIONS.                                  GMASØ58Ø
C           MTYPE = 2: CONSTANT COEFFICIENT - DIRICHLET, NEUMANN,       GMASØ59Ø
C           OR MIXED BOUNDARY CONDITIONS.                               GMASØ6ØØ
C           MTYPE = 3: GENERAL SEPARABLE - PERIODIC BOUNDARY CONDITIONS.GMASØ61Ø
C           MTYPE = 4: CONSTANT COEFFICIENT - PERIODIC BOUNDARY         GMASØ62Ø
C           CONDITIONS.                                                 GMASØ63Ø
C           RESTRICTIONS ON TM1 AND TM2 ARE ANALOGOUS TO THOSE ON       GMASØ64Ø
C           TN1 AND TN2, RESPECTIVELY, FOR EACH GIVEN TYPE.             GMASØ65Ø
C           IF M IS LESS THAN 6, GMAS MAY RESPECIFY MTYPE.              GMASØ66Ø
C       TM1 IS AN ARRAY OF LENGTH M+1, DEFINED ABOVE.                   GMASØ67Ø
C       TM2 IS AN ARRAY OF LENGTH M+1, DEFINED ABOVE.                   GMASØ68Ø
C       IBDIM IS THE ROW DIMENSION OF THE ARRAY B, AS IT APPEARS        GMASØ69Ø
C           IN THE CALLING PROGRAM.                                     GMASØ7ØØ
C       B IS A TWO DIMENSIONAL ARRAY WITH AT LEAST M ROWS AND N         GMASØ71Ø
C           COLUMNS. ON INPUT, B CONTAINS THE RIGHT HAND SIDE B(I,J)    GMASØ72Ø
C           AS DEFINED ABOVE.  ON OUTPUT, FROM A CALL TO GMAS WITH      GMASØ73Ø
C           NPC = 1, B CONTAINS THE SOLUTION X(I,J), AS DEFINED ABOVE.  GMASØ74Ø
C       VN IS AN ARRAY OF LENGTH N+1.  ON OUTPUT FROM A CALL TO GMAS    GMASØ75Ø
C           WITH NPC = Ø, VN CONTAINS THE EIGENVECTOR OF THE            GMASØ76Ø
C           MATRIX TN CORRESPONDING TO THE ZERO EIGENVALUE.             GMASØ77Ø
C       VM IS AN ARRAY OF LENGTH M+1.  ON OUTPUT FROM A CALL TO GMAS    GMASØ78Ø
C           WITH NPC = Ø, VM CONTAINS THE EIGENVECTOR OF THE            GMASØ79Ø
C           MATRIX TM CORRESPONDING TO THE ZERO EIGENVALUE.             GMASØ8ØØ
C       K IS THE MARCHING PARAMETER.  MARCHING OCCURS IN THE            GMASØ81Ø
C           N - DIRECTION.  ON INPUT, IF K IS LESS THAN 2, IT           GMASØ82Ø
C           ASSUMES THE DEFAULT VALUE K = 2.                            GMASØ83Ø
C       R IS AN ARRAY OF LENGTH N * P + 3 * N + 2, WHERE P IS AN        GMASØ84Ø
C           INTEGER GREATER THAN OR EQUAL TO LOG2( N / K ), AND K IS    GMASØ85Ø
C           GREATER THAN OR EQUAL TO 2.  R CONTAINS OUTPUT FROM A       GMASØ86Ø
C           CALL TO GMAS WITH NPC = Ø.                                  GMASØ87Ø
```

```
C       A IS AN ARRAY OF LENGTH 5 * Q, WHERE Q = MAX( N+1, M+1 ).       GMASØ88Ø
C           A IS USED AS A SCRATCHPAD ARRAY BY GMAS.                    GMASØ89Ø
C                                                                       GMASØ9ØØ
C       NOTE THAT NPC, N, NTYPE, TN1, TN2, M, MTYPE, TM1, TM2  IBDIM,   GMASØ91Ø
C           B, K, AND R ARE IDENTICAL TO THE CORRESPONDING PARAMETERS   GMASØ92Ø
C           IN THE CALLING SEQUENCE FOR SUBROUTINE GMA.                 GMASØ93Ø
C           THE PARAMETERS VN, VM, AND A ARE DIFFERENT.                 GMASØ94Ø
C                                                                       GMASØ95Ø
C           GMAS CONTAINS ONE LABELED COMMON BLOCK, /MACHEP/,           GMASØ96Ø
C       CONTAINING ON VARIABLE, TOL.  TOL IS A MACHINE DEPENDENT        GMASØ97Ø
C       CONSTANT EQUAL TO THE MACHINE EPSILON.  IT IS INITIALIZED IN    GMASØ98Ø
C       GMAS IN THE FIRST EXECUTABLE STATEMENT.                         GMASØ99Ø
C                                                                       GMAS1ØØØ
C           A CALL TO GMAS WITH NPC = Ø REPLACES A CALL TO GMA WITH     GMAS1Ø1Ø
C       NPC = Ø.                                                        GMAS1Ø2Ø
C                                                                       GMAS1Ø3Ø
C           A CALL TO GMAS WITH NPC = 1 REPLACES A CALL TO GMA WITH     GMAS1Ø4Ø
C       WITH NPC = 1.  THE RIGHT HAND SIDE B IS PROJECTED INTO THE      GMAS1Ø5Ø
C       SUBSPACE ORTHOGONAL TO THE VECTOR V, WHERE V(I,J) =             GMAS1Ø6Ø
C       VM(I) * VN(J).  THIS IS NECESSARY TO INSURE THAT THE LINEAR     GMAS1Ø7Ø
C       SYSTEM IS CONSISTENT.  THE SOLUTION X IS ALSO PROJECTED INTO    GMAS1Ø8Ø
C       THE SUBSPACE ORTHOGONAL TO V.  ALL OTHER SOLUTIONS CAN BE       GMAS1Ø9Ø
C       OBTAINED BY ADDING AN ARBITRARY SCALAR MULTIPLE OF V TO X.      GMAS11ØØ
C                                                                       GMAS111Ø
C       ADDRESS INQUIRIES TO:                                           GMAS112Ø
C           RANDOLPH E. BANK                                            GMAS113Ø
C           DEPARTMENT OF MATHEMATICS                                   GMAS114Ø
C           THE UNIVERSITY OF CHICAGO                                   GMAS115Ø
C           CHICAGO, ILLINOIS 6Ø637                                     GMAS116Ø
C                                                                       GMAS117Ø
C       VERSION DATE: MARCH 1, 1977.                                    GMAS118Ø
            DIMENSION TN1(1),TN2(1),TM1(1),TM2(1)                       GMAS119Ø
            DIMENSION VN(1),VM(1),A(1),B(1),R(1)                        GMAS12ØØ
            COMMON /MACHEP/TOL                                          GMAS121Ø
            TOL=2.ØEØ**(-23)                                            GMAS122Ø
C           CHECK N, NTYPE, M, MTYPE, AND IBDIM.                        GMAS123Ø
            IF((NTYPE.GT.4).OR.(NTYPE.LT.1)) RETURN                     GMAS124Ø
            IF((MTYPE.GT.4).OR.(MTYPE.LT.1)) RETURN                     GMAS125Ø
            IF((N.LT.2).OR.(M.LT.2)) RETURN                             GMAS126Ø
            IF(IBDIM.LT.M) RETURN                                       GMAS127Ø
C           RESPECIFY NTYPE AND/OR MTYPE IF NECESSARY.                  GMAS128Ø
            IF(NTYPE.EQ.2.AND.N.LE.3) NTYPE=1                           GMAS129Ø
            IF(MTYPE.EQ.2.AND.M.LE.3) MTYPE=1                           GMAS13ØØ
            IF(NTYPE.EQ.4.AND.N.LE.3) NTYPE=3                           GMAS131Ø
            IF(MTYPE.EQ.4.AND.M.LE.5) MTYPE=3                           GMAS132Ø
            IF(N.LE.2) NTYPE=1                                          GMAS133Ø
            IF(M.LE.2) MTYPE=1                                          GMAS134Ø
            IF(NPC.NE.Ø) GO TO 15                                       GMAS135Ø
C           THE PREPROCESSING CALCULATIONS.                             GMAS136Ø
            KK=MINØ(K,N)                                                GMAS137Ø
            CALL PARTN(N,NTYPE,KK,NE,ND,R(3))                           GMAS138Ø
C           EPS IS ADDED TO ALL INTEGERS STORED IN THE REAL ARRAY R.    GMAS139Ø
C           THIS PREVENTS UNFORTUNATE ROUNDING ERRORS FORM OCCURRING WHEN GMAS14ØØ
C           THEY ARE CONVERTED BACK INTO INTEGERS.                      GMAS141Ø
            EPS=Ø.25EØ                                                  GMAS142Ø
            R(1)=FLOAT(NE)+EPS                                          GMAS143Ø
            R(2)=FLOAT(ND)+EPS                                          GMAS144Ø
            IF(NTYPE.EQ.1.OR.NTYPE.EQ.3) CALL ROOTSG(N,NTYPE,           GMAS145Ø
         1      TN1,TN2,NE,ND,R(3),R(N+3),A)                           GMAS146Ø
            IF(NTYPE.EQ.2.OR.NTYPE.EQ.4) CALL ROOTSC(N,NTYPE,           GMAS147Ø
         1      TN1,TN2,NE,ND,R(3),R(N+3),A)                           GMAS148Ø
            CALL SVALUE(N,NTYPE,M,MTYPE,TM1,TM2,NE,R(N+3),TMIN)         GMAS149Ø
            J1=N+2                                                      GMAS15ØØ
            J2=J1+N+1                                                   GMAS151Ø
            CALL SORT(N,NTYPE,NE,ND,R(3),R(N+3),A(1),A(J1),A(J2))       GMAS152Ø
            J=MAXØ(N,M)+1                                               GMAS153Ø
            J1=J+1                                                      GMAS154Ø
            J2=J1+J                                                     GMAS155Ø
            J3=J2+J                                                     GMAS156Ø
            J4=J3+J                                                     GMAS157Ø
            IF(NTYPE.LT.3) CALL TINVIT(N,TN1,TN2,TMIN,VN,               GMAS158Ø
         1      A(1),A(J1),A(J2),A(J3),IFLAG)                          GMAS159Ø
            IF(NTYPE.GE.3) CALL PINVIT(N,TN1,TN2,TMIN,VN,               GMAS16ØØ
         1      A(1),A(J1),A(J2),A(J3),A(J4),IFLAG)                    GMAS161Ø
            TMIN=-TMIN                                                  GMAS162Ø
            IF(MTYPE.LT.3) CALL TINVIT(M,TM1,TM2,TMIN,VM,               GMAS163Ø
```

```
     1      A(1),A(J1),A(J2),A(J3),IFLAG)                                  GMAS164Ø
            IF(MTYPE.GE.3) CALL PINVIT(M,TM1,TM2,TMIN,VM,                  GMAS165Ø
     1      A(1),A(J1),A(J2),A(J3),A(J4),IFLAG)                            GMAS166Ø
            RETURN                                                         GMAS167Ø
C           THE BACKSOLUTION CALCULATIONS.                                 GMAS168Ø
   15       TN2(N+1)=1.ØEØ                                                 GMAS169Ø
            IF(NTYPE.LT.3) TN2(1)=1.ØEØ                                    GMAS17ØØ
            NE=R(1)                                                        GMAS171Ø
            ND=R(2)                                                        GMAS172Ø
            CALL PROJ(1,N,NTYPE,VN,M,VM,IBDIM,B,NE,ND,R(3),COEFF)          GMAS173Ø
            CALL STEP1(N,NTYPE,TN1,TN2,M,MTYPE,TM1,TM2,                    GMAS174Ø
     1      IBDIM,B,ND,R(3),R(N+3),A)                                      GMAS175Ø
            CALL PROJ(2,N,NTYPE,VN,M,VM,IBDIM,B,NE,ND,R(3),COEFF)          GMAS176Ø
            CALL STEP2(N,NTYPE,TN2,M,MTYPE,TM1,TM2,                        GMAS177Ø
     1      IBDIM,B,NE,ND,R(3),R(N+3),A)                                   GMAS178Ø
            CALL PROJ(3,N,NTYPE,VN,M,VM,IBDIM,B,NE,ND,R(3),COEFF)          GMAS179Ø
            CALL STEP3(N,NTYPE,TN2,M,MTYPE,TM1,TM2,                        GMAS18ØØ
     1      IBDIM,B,NE,ND,R(3),R(N+3),A)                                   GMAS181Ø
            CALL STEP4(TN1,TN2,M,MTYPE,TM1,TM2,IBDIM,B,ND,R(3),R(N+3),A)   GMAS182Ø
            CALL PROJ(1,N,NTYPE,VN,M,VM,IBDIM,B,NE,ND,R(3),COEFF)          GMAS183Ø
            RETURN                                                         GMAS184Ø
            END                                                            GMAS185Ø
C
C
            SUBROUTINE SVALUE(N,NTYPE,M,MTYPE,TM1,TM2,NE,R,TMIN)           SVALØØ1Ø
C               SUBROUTINE SVALUE DETERMINES TMIN, THE ROOT ASSOCIATED     SVALØØ2Ø
C           WITH THE ZERO EIGENVALUE.  THIS ROOT IS PERTURBED TO PREVENT   SVALØØ3Ø
C           POSSIBLE DIVIDE CHECKS DURING THE BACKSOLUTION PROCESS.        SVALØØ4Ø
C           VERSION DATE: FEBRUARY 1, 1977.                                SVALØØ5Ø
                DIMENSION TM1(1),TM2(1),R(1)                               SVALØØ6Ø
                COMMON /MACHEP/TOL                                         SVALØØ7Ø
            IF(MTYPE.LT.3) TM2(1)=Ø.ØEØ                                    SVALØØ8Ø
            TM2(M+1)=TM2(1)                                                SVALØØ9Ø
            T=Ø.ØEØ                                                        SVALØ1ØØ
            Q=ABS(TM2(1))                                                  SVALØ11Ø
            DO 5 I=1,M                                                     SVALØ12Ø
                S=Q                                                        SVALØ13Ø
                Q=ABS(TM2(I+1))                                            SVALØ14Ø
                T=AMAX1(T,ABS(TM1(I))+Q+S)                                 SVALØ15Ø
    5           CONTINUE                                                   SVALØ16Ø
            L=N*NE                                                         SVALØ17Ø
            IF(NTYPE.GE.3) L=L+N                                           SVALØ18Ø
            JMIN=L+1                                                       SVALØ19Ø
            IF((TM1(1)+R(JMIN)).LT.Ø.ØEØ) T=-T                             SVALØ2ØØ
            L=L+N                                                          SVALØ21Ø
            IF(ABS(T+R(JMIN)).GT.ABS(T+R(L))) JMIN=L                       SVALØ22Ø
            TMIN=R(JMIN)                                                   SVALØ23Ø
            R(JMIN)=TMIN+(T+TMIN)*FLOAT(M)*TOL                             SVALØ24Ø
            RETURN                                                         SVALØ25Ø
            END                                                            SVALØ26Ø
C
C
            SUBROUTINE TINVIT(N,D,E,EIGEN,V,A,B,C,F,IFLAG)                 TINVØØ1Ø
C               SUBROUTINE TINVIT COMPUTES THE EIGENVECTOR OF A            TINVØØ2Ø
C           SYMMETRIC TRIDIAGONAL MATRIX CORRESPONDING TO THE EIGENVALUE   TINVØØ3Ø
C           EIGEN. INVERSE ITERATION AND GAUSSIAN ELIMINATION WITH         TINVØØ4Ø
C           PARTIAL PIVOTING IS EMPLOYED.  D AND E CONTAIN THE             TINVØØ5Ø
C           TRIDIAGONAL MATRIX.  A, B, AND C CONTAIN THE UPPER TRIANGULAR  TINVØØ6Ø
C           MATRIX.  F CONTAINS THE LOWER TRIANGULAR MATRIX.               TINVØØ7Ø
C           V CONTAINS THE EIGENVECTOR.                                    TINVØØ8Ø
C           VERSION DATE: FEBRUARY 1, 1977.                                TINVØØ9Ø
                DIMENSION A(1),B(1),C(1),D(1),E(1),F(1),V(1)               TINVØ1ØØ
                COMMON /MACHEP/TOL                                         TINVØ11Ø
            IFLAG=Ø                                                        TINVØ12Ø
            Q=ABS(D(1))                                                    TINVØ13Ø
            DO 5 I=2,N                                                     TINVØ14Ø
    5           Q=Q+ABS(D(I))+ABS(E(I))                                    TINVØ15Ø
            EPS=TOL*Q*FLOAT(N)                                             TINVØ16Ø
            Z=TOL*Q*SQRT(FLOAT(N))                                         TINVØ17Ø
            E(N+1)=Ø.ØEØ                                                   TINVØ18Ø
            R=E(2)                                                         TINVØ19Ø
            U=D(1)-EIGEN                                                   TINVØ2ØØ
            V(1)=Z                                                         TINVØ21Ø
            DO 15 I=2,N                                                    TINVØ22Ø
                IM1=I-1                                                    TINVØ23Ø
                V(I)=Z                                                     TINVØ24Ø
```

```
            IF(ABS(E(I)).GT.ABS(U)) GO TO 1Ø                          TINVØ25Ø
            Q=E(I)/U                                                  TINVØ26Ø
            F(I)=Q                                                    TINVØ27Ø
            A(IM1)=U                                                  TINVØ28Ø
            B(IM1)=R                                                  TINVØ29Ø
            C(IM1)=Ø.ØEØ                                              TINVØ3ØØ
            U=D(I)-EIGEN-Q*R                                          TINVØ31Ø
            R=E(I+1)                                                  TINVØ32Ø
            GO TO 15                                                  TINVØ33Ø
   1Ø       Q=U/E(I)                                                  TINVØ34Ø
            F(I)=Q                                                    TINVØ35Ø
            A(IM1)=-E(I)                                              TINVØ36Ø
            B(IM1)=EIGEN-D(I)                                         TINVØ37Ø
            C(IM1)=E(I+1)                                             TINVØ38Ø
            U=-R-Q*B(IM1)                                             TINVØ39Ø
            R=Q*C(IM1)                                                TINVØ4ØØ
   15       CONTINUE                                                  TINVØ41Ø
        IF(U.EQ.Ø.ØEØ) U=EPS/FLOAT(N)                                 TINVØ42Ø
        A(N)=U                                                        TINVØ43Ø
        B(N)=Ø.ØEØ                                                    TINVØ44Ø
        C(N)=Ø.ØEØ                                                    TINVØ45Ø
        L=1                                                           TINVØ46Ø
        DO 5Ø J=1,5                                                   TINVØ47Ø
            DO 25 I=2,N                                               TINVØ48Ø
                IM1=I-1                                               TINVØ49Ø
                Q=V(I)                                                TINVØ5ØØ
                R=-E(I)                                               TINVØ51Ø
                IF(A(IM1).NE.R) GO TO 25                              TINVØ52Ø
                Q=V(IM1)                                              TINVØ53Ø
                V(IM1)=V(I)                                           TINVØ54Ø
   25           V(I)=Q+F(I)*V(IM1)                                    TINVØ55Ø
            Q=Ø.ØEØ                                                   TINVØ56Ø
            R=Ø.ØEØ                                                   TINVØ57Ø
            Z=Ø.ØEØ                                                   TINVØ58Ø
            DO 3Ø II=1,N                                              TINVØ59Ø
                I=N+1-II                                              TINVØ6ØØ
                V(I)=(V(I)+Q*B(I)+R*C(I))/A(I)                        TINVØ61Ø
                R=Q                                                   TINVØ62Ø
                Q=V(I)                                                TINVØ63Ø
   3Ø           Z=Z+ABS(Q)                                            TINVØ64Ø
            IF(Z.GT.1.ØEØ) GO TO 6Ø                                   TINVØ65Ø
            IF(Z.NE.Ø.ØEØ) GO TO 4Ø                                   TINVØ66Ø
            V(L)=EPS                                                  TINVØ67Ø
            L=L+1                                                     TINVØ68Ø
            IF(L.GT.N) L=1                                            TINVØ69Ø
            GO TO 5Ø                                                  TINVØ7ØØ
   4Ø       Z=EPS/Z                                                   TINVØ71Ø
            DO 45 I=1,N                                               TINVØ72Ø
   45           V(I)=V(I)*Z                                           TINVØ73Ø
   5Ø       CONTINUE                                                  TINVØ74Ø
        IFLAG=1                                                       TINVØ75Ø
   6Ø   Z=Ø.ØEØ                                                       TINVØ76Ø
        DO 65 I=1,N                                                   TINVØ77Ø
   65       Z=Z+V(I)*V(I)                                             TINVØ78Ø
        Z=1.ØEØ/SQRT(Z)                                               TINVØ79Ø
        DO 7Ø I=1,N                                                   TINVØ8ØØ
   7Ø       V(I)=V(I)*Z                                               TINVØ81Ø
        RETURN                                                        TINVØ82Ø
        END                                                           TINVØ83Ø
C
C
        SUBROUTINE PINVIT(N,D,E,EIGEN,V,A,B,C,F,G,IFLAG)              PINVØØ1Ø
C           SUBROUTINE PINVIT COMPUTES THE EIGENVECTOR OF A           PINVØØ2Ø
C       SYMMETRIC MATRIX ASSOCIATED WITH PERIODIC BOUNDARY CONDITIONS, PINVØØ3Ø
C       CORRESPONDING THE THE EIGENVALUE EIGEN.  INVERSE ITERATION    PINVØØ4Ø
C       AND GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING ARE EMPLOYED.  PINVØØ5Ø
C       D AND E CONTAIN THE MATRIX WITH PERIODIC BOUNDARY CONDITIONS. PINVØØ6Ø
C       A, B, C, AND G CONTAIN THE UPPER TRIANGULAR MATRIX.           PINVØØ7Ø
C       F CONTAINS THE LOWER TRIANGULAR MATRIX.  V CONTAINS THE       PINVØØ8Ø
C       EIGENVECTOR.                                                  PINVØØ9Ø
C       VERSION DATE: FEBRUARY 1, 1977.                               PINVØ1ØØ
            DIMENSION A(1),B(1),C(1),D(1),E(1),F(1),G(1),V(1)         PINVØ11Ø
            COMMON /MACHEP/TOL                                        PINVØ12Ø
        IFLAG=Ø                                                       PINVØ13Ø
        NM1=N-1                                                       PINVØ14Ø
        Q=Ø.ØEØ                                                       PINVØ15Ø
```

```
      DO 5 I=1,N                                                    PINVØ16Ø
          G(I)=Ø.ØEØ                                                PINVØ17Ø
 5        Q=Q+ABS(D(I))+ABS(E(I))                                   PINVØ18Ø
      EPS=TOL*Q*FLOAT(N)                                            PINVØ19Ø
      Z=TOL*Q*SQRT(FLOAT(N))                                        PINVØ2ØØ
      R=E(2)                                                        PINVØ21Ø
      U=D(1)-EIGEN                                                  PINVØ22Ø
      V(1)=Z                                                        PINVØ23Ø
      V(N)=Z                                                        PINVØ24Ø
      G(1)=E(1)                                                     PINVØ25Ø
      G(NM1)=E(N)                                                   PINVØ26Ø
      DO 15 I=2,NM1                                                 PINVØ27Ø
          IM1=I-1                                                   PINVØ28Ø
          V(I)=Z                                                    PINVØ29Ø
          IF(ABS(E(I)).GT.ABS(U)) GO TO 1Ø                          PINVØ3ØØ
          Q=E(I)/U                                                  PINVØ31Ø
          F(I)=Q                                                    PINVØ32Ø
          A(IM1)=U                                                  PINVØ33Ø
          B(IM1)=R                                                  PINVØ34Ø
          C(IM1)=Ø.ØEØ                                              PINVØ35Ø
          U=D(I)-EIGEN-Q*R                                          PINVØ36Ø
          R=E(I+1)                                                  PINVØ37Ø
          G(I)=G(I)+Q*G(IM1)                                        PINVØ38Ø
          GO TO 15                                                  PINVØ39Ø
 1Ø       Q=U/E(I)                                                  PINVØ4ØØ
          F(I)=Q                                                    PINVØ41Ø
          A(IM1)=-E(I)                                              PINVØ42Ø
          B(IM1)=EIGEN-D(I)                                         PINVØ43Ø
          C(IM1)=E(I+1)                                             PINVØ44Ø
          U=-R-Q*B(IM1)                                             PINVØ45Ø
          R=G(IM1)                                                  PINVØ46Ø
          G(IM1)=G(I)                                               PINVØ47Ø
          G(I)=R+Q*G(IM1)                                           PINVØ48Ø
          R=Q*C(IM1)                                                PINVØ49Ø
 15       CONTINUE                                                  PINVØ5ØØ
      IF(U.EQ.Ø.ØEØ) U=EPS/FLOAT(N)                                 PINVØ51Ø
      A(NM1)=U                                                      PINVØ52Ø
      B(NM1)=Ø.ØEØ                                                  PINVØ53Ø
      C(NM1)=Ø.ØEØ                                                  PINVØ54Ø
      C(N-2)=Ø.ØEØ                                                  PINVØ55Ø
      Q=Ø.ØEØ                                                       PINVØ56Ø
      R=Ø.ØEØ                                                       PINVØ57Ø
      DO 2Ø II=1,NM1                                                PINVØ58Ø
          I=N-II                                                    PINVØ59Ø
          G(I)=(G(I)+Q*B(I)+R*C(I))/A(I)                            PINVØ6ØØ
          R=Q                                                       PINVØ61Ø
 2Ø       Q=G(I)                                                    PINVØ62Ø
      U=D(N)-EIGEN-E(1)*G(1)-E(N)*G(NM1)                            PINVØ63Ø
      IF(U.EQ.Ø.ØEØ) U=EPS/FLOAT(N)                                 PINVØ64Ø
      A(N)=U                                                        PINVØ65Ø
      L=1                                                           PINVØ66Ø
      DO 5Ø J=1,5                                                   PINVØ67Ø
          DO 25 I=2,NM1                                             PINVØ68Ø
              IM1=I-1                                               PINVØ69Ø
              Q=V(I)                                                PINVØ7ØØ
              R=-E(I)                                               PINVØ71Ø
              IF(A(IM1).NE.R) GO TO 25                              PINVØ72Ø
              Q=V(IM1)                                              PINVØ73Ø
              V(IM1)=V(I)                                           PINVØ74Ø
 25           V(I)=Q+F(I)*V(IM1)                                    PINVØ75Ø
          Q=Ø.ØEØ                                                   PINVØ76Ø
          R=Ø.ØEØ                                                   PINVØ77Ø
          DO 3Ø II=1,NM1                                            PINVØ78Ø
              I=N-II                                                PINVØ79Ø
              V(I)=(V(I)+Q*B(I)+R*C(I))/A(I)                        PINVØ8ØØ
              R=Q                                                   PINVØ81Ø
 3Ø           Q=V(I)                                                PINVØ82Ø
          Q=(V(N)+E(1)*V(1)+E(N)*V(NM1))/A(N)                       PINVØ83Ø
          V(N)=Q                                                    PINVØ84Ø
          Z=ABS(Q)                                                  PINVØ85Ø
          DO 35 I=1,NM1                                             PINVØ86Ø
              V(I)=V(I)+G(I)*Q                                      PINVØ87Ø
 35           Z=Z+ABS(V(I))                                         PINVØ88Ø
          IF(Z.GT.1.ØEØ) GO TO 6Ø                                   PINVØ89Ø
          IF(Z.NE.Ø.ØEØ) GO TO 4Ø                                   PINVØ9ØØ
          V(L)=EPS                                                  PINVØ91Ø
```

```
            L=L+1                                                          PINV0920
            IF(L.GT.N) L=1                                                 PINV0930
            GO TO 50                                                       PINV0940
   40       Z=EPS/Z                                                        PINV0940
            DO 45 I=1,N                                                    PINV0950
   45           V(I)=V(I)*Z                                               PINV0960
   50       CONTINUE                                                       PINV0970
        IFLAG=1                                                            PINV0980
   60   Z=0.0E0                                                           PINV0990
        DO 65 I=1,N                                                        PINV1000
   65       Z=Z+V(I)*V(I)                                                 PINV1010
        Z=1.0E0/SQRT(Z)                                                    PINV1020
        DO 70 I=1,N                                                        PINV1030
   70       V(I)=V(I)*Z                                                   PINV1040
        RETURN                                                             PINV1050
        END                                                                PINV1060
C                                                                          PINV1070
C
        SUBROUTINE PROJ(ITYPE,N,NTYPE,VN,M,VM,IBDIM,B,NE,ND,ID,COEFF)      PROJ0010
C           SUBROUTINE PROJ COMPUTES THE PROJECTIONS NECESSARY TO          PROJ0020
C       INSURE THAT THE LINEAR SYSTEM IS CONSISTENT.                       PROJ0030
C       VERSION DATE: FEBRUARY 1, 1977.                                    PROJ0040
            REAL ID                                                        PROJ0050
            DIMENSION VN(1),VM(1),B(1),ID(1)                               PROJ0060
        COEFF=0.0E0                                                        PROJ0070
        GO TO (5,30,55),ITYPE                                              PROJ0080
    5   DO 15 I=1,N                                                        PROJ0090
            T=0.0E0                                                        PROJ0100
            LNPTR=(I-1)*IBDIM                                              PROJ0110
            D                                                              PROJ0120
             O 10 J=1,M                                                    PROJ0130
                LNPTR=LNPTR+1                                              PROJ0140
   10           T=T+VM(J)*B(LNPTR)                                         PROJ0150
   15       COEFF=COEFF+VN(I)*T                                           PROJ0160
        DO 25 I=1,N                                                        PROJ0170
            T=COEFF*VN(I)                                                  PROJ0180
            LNPTR=(I-1)*IBDIM                                              PROJ0190
            DO 20 J=1,M                                                    PROJ0200
                LNPTR=LNPTR+1                                              PROJ0210
   20           B(LNPTR)=B(LNPTR)-T*VM(J)                                  PROJ0220
   25       CONTINUE                                                       PROJ0230
        RETURN                                                             PROJ0240
   30   NN=ND                                                             PROJ0250
        IF(NTYPE.GE.3) NN=ND+1                                             PROJ0260
        ID2=ID(2)                                                          PROJ0270
        IF((NN.LE.1).OR.(ID2.EQ.2)) RETURN                                 PROJ0280
        Q=0.0E0                                                            PROJ0290
        DO 40 I=1,NN                                                       PROJ0300
            LINE=ID(I+1)                                                   PROJ0310
            T=0.0E0                                                        PROJ0320
            LNPTR=(LINE-1)*IBDIM                                           PROJ0330
            DO 35 J=1,M                                                    PROJ0340
                LNPTR=LNPTR+1                                              PROJ0350
   35           T=T+VM(J)*B(LNPTR)                                         PROJ0360
            Q=Q+VN(LINE)*VN(LINE)                                          PROJ0370
   40       COEFF=COEFF+VN(LINE)*T                                        PROJ0380
        COEFF=COEFF/Q                                                      PROJ0390
        DO 50 I=1,NN                                                       PROJ0400
            LINE=ID(I+1)                                                   PROJ0410
            T=COEFF*VN(LINE)                                               PROJ0420
            LNPTR=(LINE-1)*IBDIM                                           PROJ0430
            DO 45 J=1,M                                                    PROJ0440
                LNPTR=LNPTR+1                                              PROJ0450
   45           B(LNPTR)=B(LNPTR)-T*VM(J)                                  PROJ0460
   50       CONTINUE                                                       PROJ0470
        RETURN                                                             PROJ0480
   55   NN=NE-1                                                           PROJ0490
        IF(NTYPE.GE.3) NN=NE                                               PROJ0500
        IF(NN.LT.0) RETURN                                                 PROJ0510
        I=2**NN+1                                                          PROJ0520
        I=ID(I)                                                            PROJ0530
        LNPTR=(I-1)*IBDIM                                                  PROJ0540
        DO 60 J=1,M                                                        PROJ0550
            LNPTR=LNPTR+1                                                  PROJ0560
   60       COEFF=COEFF+VM(J)*B(LNPTR)                                    PROJ0570
        LNPTR=(I-1)*IBDIM                                                  PROJ0580
```

```
      DO 65 J=1,M                                                        PROJØ59Ø
          LNPTR=LNPTR+1                                                  PROJØ6ØØ
   65     B(LNPTR)=B(LNPTR)-COEFF*VM(J)                                  PROJØ61Ø
      RETURN                                                             PROJØ62Ø
      END
C
C
      SUBROUTINE EIGEN(N,NTYPE,D,E,K,EIG)                                EIGNØØ1Ø
C         SUBROUTINE EIGEN COMPUTES THE K TH EIGENVALUE FOR              EIGNØØ2Ø
C     MATRICES OF THE TYPE ALLOWED BY SUBROUTINE GMA.                    EIGNØØ3Ø
C     VERSION DATE: FEBRUARY 1, 1977.                                    EIGNØØ4Ø
          DIMENSION D(1),E(1)                                            EIGNØØ5Ø
      PI=3.14159265358979323846264338327952EØ                            EIGNØØ6Ø
      E(N+1)=E(N)                                                        EIGNØØ7Ø
      IF(NTYPE.LT.3) E(1)=E(2)                                           EIGNØØ8Ø
      GO TO (1Ø,15,5Ø,6Ø),NTYPE                                          EIGNØØ9Ø
   1Ø CALL TRIEIG(N,D,E,K,EIG)                                           EIGNØ1ØØ
      RETURN                                                             EIGNØ11Ø
   15 IF(N.LE.3) GO TO 1Ø                                                EIGNØ12Ø
      I=15                                                               EIGNØ13Ø
      T1=D(2)                                                            EIGNØ14Ø
      T2=E(3)                                                            EIGNØ15Ø
      S2=SQRT(2.ØEØ)*T2                                                  EIGNØ16Ø
      Q1=D(1)                                                            EIGNØ17Ø
      Q2=E(2)                                                            EIGNØ18Ø
      IF(Q1.EQ.T1.AND.Q2.EQ.T2) I=I-7                                    EIGNØ19Ø
      IF(Q1.EQ.T1.AND.Q2.EQ.S2) I=I-6                                    EIGNØ2ØØ
      IF(Q1.EQ.(T1-T2).AND.Q2.EQ.T2) I=I-4                               EIGNØ21Ø
      Q1=D(N)                                                            EIGNØ22Ø
      Q2=E(N)                                                            EIGNØ23Ø
      IF(Q1.EQ.T1.AND.Q2.EQ.T2) I=I-7                                    EIGNØ24Ø
      IF(Q1.EQ.T1.AND.Q2.EQ.S2) I=I-6                                    EIGNØ25Ø
      IF(Q1.EQ.(T1-T2).AND.Q2.EQ.T2) I=I-4                               EIGNØ26Ø
      IF(I.GT.7) GO TO 1Ø                                                EIGNØ27Ø
      GO TO (2Ø,25,3Ø,35,4Ø,1Ø,45),I                                     EIGNØ28Ø
   2Ø EIG=T1-T2*2.ØEØ*COS(FLOAT(K)*PI/FLOAT(N+1))                        EIGNØ29Ø
      RETURN                                                             EIGNØ3ØØ
   25 EIG=T1-T2*2.ØEØ*COS(FLOAT(2*K-1)*PI/FLOAT(2*N))                    EIGNØ31Ø
      RETURN                                                             EIGNØ32Ø
   3Ø EIG=T1-T2*2.ØEØ*COS(FLOAT(K-1)*PI/FLOAT(N-1))                      EIGNØ33Ø
      RETURN                                                             EIGNØ34Ø
   35 EIG=T1-T2*2.ØEØ*COS(FLOAT(2*K-1)*PI/FLOAT(2*N+1))                  EIGNØ35Ø
      RETURN                                                             EIGNØ36Ø
   4Ø EIG=T1-T2*2.ØEØ*COS(FLOAT(2*(K-1))*PI/FLOAT(2*N-1))                EIGNØ37Ø
      RETURN                                                             EIGNØ38Ø
   45 EIG=T1-T2*2.ØEØ*COS(FLOAT(K-1)*PI/FLOAT(N))                        EIGNØ39Ø
      RETURN                                                             EIGNØ4ØØ
   5Ø CALL PEREIG(N,D,E,K,EIG)                                           EIGNØ41Ø
      RETURN                                                             EIGNØ42Ø
   6Ø I=(K/2)*2                                                          EIGNØ43Ø
      EIG=D(1)-E(3)*2.ØEØ*COS(FLOAT(I)*PI/FLOAT(N))                      EIGNØ44Ø
      RETURN                                                             EIGNØ45Ø
      END                                                                EIGNØ46Ø
C
C
      SUBROUTINE ERROR(EMARCH,BIG,TMMAX,N,TN1,TN2,K,ND)                  ERORØØ1Ø
C         SUBROUTINE ERROR PARTITIONS THE LINEAR SYSTEM USING THE        ERORØØ2Ø
C     SAME ALGORITHM AS SUBROUTINE GMA.  EMARCH IS DETERMINED BY         ERORØØ3Ø
C     USING A ONE DIMENSIONAL MARCH WITH THE SCALAR TMMAX PLAYING THE    ERORØØ4Ø
C     ROLE OF THE MATRIX TM.                                             ERORØØ5Ø
C     VERSION DATE: FEBRUARY 1, 1977.                                    ERORØØ6Ø
          DIMENSION TN1(1),TN2(1)                                        ERORØØ7Ø
      EMARCH=Ø.ØEØ                                                       ERORØØ8Ø
      ND=N/MAXØ(2,K)                                                     ERORØØ9Ø
      IF(ND*2.EQ.N) ND=ND-1                                              ERORØ1ØØ
      NDP1=ND+1                                                          ERORØ11Ø
      I1=N/NDP1                                                          ERORØ12Ø
      I2=N-I1*NDP1                                                       ERORØ13Ø
      I3=Ø                                                               ERORØ14Ø
      ID=Ø                                                               ERORØ15Ø
      DO 65 LNINDX=1,NDP1                                                ERORØ16Ø
          LINE=ID+1                                                      ERORØ17Ø
          IF(I3.LT.I2) I3=I3+1                                           ERORØ18Ø
          ID=I1*LNINDX+I3                                                ERORØ19Ø
          IF(LNINDX.EQ.NDP1) ID=N+1                                      ERORØ2ØØ
          ISPAN=ID-1                                                     ERORØ21Ø
          R1=1.ØEØ                                                       ERORØ22Ø
```

```
            R2=Ø.ØEØ                                                    ERORØ23Ø
            DO 55 J=LINE,ISPAN                                          ERORØ24Ø
                IF(ABS(R1).GT.BIG) GO TO 1ØØ                            ERORØ25Ø
                T=((TMMAX+TN1(J))*R1-TN2(J)*R2)/TN2(J+1)                ERORØ26Ø
                R2=R1                                                   EROR027Ø
   55           R1=T                                                    EROR028Ø
            R1=ABS(R1)                                                  EROR029Ø
            R2=ABS(TN2(ID)/TN2(LINE))*R1                                EROR03ØØ
            T=AMAX1(R1,R2)                                              EROR031Ø
            EMARCH=AMAX1(T,EMARCH)                                      EROR032Ø
   65       CONTINUE                                                    EROR033Ø
        RETURN                                                          EROR034Ø
  1ØØ   EMARCH=BIG                                                      EROR035Ø
        RETURN                                                          EROR036Ø
        END                                                             EROR037Ø
C
C
        SUBROUTINE TRIEIG(N,D,E,K,EIGEN)                                TEIGØØ1Ø
C           SUBROUTINE TRIEIG COMPUTES THE K TH EIGENVALUE OF AN        TEIGØØ2Ø
C       IRREDUCIBLE SYMMETRIC TRIDIAGONAL MATRIX USING BISECTION        TEIGØØ3Ø
C       AND THE STURM SEQUENCE PROPERTY.  INITIAL UPPER AND LOWER       TEIGØØ4Ø
C       BOUNDS ARE DETERMINED FROM GERSCHGORIN ESTIMATES.               TEIGØØ5Ø
C       VERSION DATE: FEBRUARY 1, 1977.                                 TEIGØØ6Ø
            DIMENSION D(1),E(1)                                         TEIGØØ7Ø
            COMMON /MACHEP/TOL                                          TEIGØØ8Ø
        E(N+1)=Ø.ØEØ                                                    TEIGØØ9Ø
        T=Ø.ØEØ                                                         TEIGØ1ØØ
        XU=D(1)                                                         TEIGØ11Ø
        XL=D(1)                                                         TEIGØ12Ø
        DO 5 I=1,N                                                      TEIGØ13Ø
            X=T                                                         TEIGØ14Ø
            T=ABS(E(I+1))                                               TEIGØ15Ø
            XU=AMAX1(XU,D(I)+(X+T))                                     TEIGØ16Ø
    5       XL=AMIN1(XL,D(I)-(X+T))                                     TEIGØ17Ø
        EPS=AMAX1(ABS(XU),ABS(XL))*TOL                                  TEIGØ18Ø
        XU=XU+EPS                                                       TEIGØ19Ø
        XL=XL-EPS                                                       TEIGØ2ØØ
        E(N+1)=E(N)                                                     TEIGØ21Ø
        E(1)=Ø.ØEØ                                                      TEIGØ22Ø
   1Ø   X=(XL+XU)/2.ØEØ                                                 TEIGØ23Ø
        T=EPS+4.ØEØ*TOL*(ABS(XL)+ABS(XU))                               TEIGØ24Ø
        IF((XU-XL).LE.T) GO TO 5Ø                                       TEIGØ25Ø
        KCOUNT=Ø                                                        TEIGØ26Ø
        T=1.ØEØ                                                         TEIGØ27Ø
        DO 2Ø I=1,N                                                     TEIGØ28Ø
            IF(T.EQ.Ø.ØEØ) T=TOL*ABS(E(I))                             TEIGØ29Ø
            T=D(I)-X-E(I)*E(I)/T                                        TEIGØ3ØØ
            IF(T.LT.Ø.ØEØ) KCOUNT=KCOUNT+1                             TEIGØ31Ø
   2Ø       CONTINUE                                                    TEIGØ32Ø
        IF(KCOUNT-K) 3Ø,35,35                                           TEIGØ33Ø
   3Ø   XL=X                                                            TEIGØ34Ø
        GO TO 1Ø                                                        TEIGØ35Ø
   35   XU=X                                                            TEIGØ36Ø
        GO TO 1Ø                                                        TEIGØ37Ø
   5Ø   EIGEN=X                                                         TEIGØ38Ø
        E(1)=E(2)                                                       TEIGØ39Ø
        RETURN                                                          TEIGØ4ØØ
        END
C
C
        SUBROUTINE PEREIG(N,D,E,K,EIGEN)                                PEIGØØ1Ø
C           SUBROUTINE PEREIG COMPUTES THE K TH EIGENVALUE OF THE       PEIGØØ2Ø
C       IRREDUCIBLE SYMMETRIC MATRIX WHICH ARISES IN THE CASE OF        PEIGØØ3Ø
C       PERIODIC BOUNDARY CONDITIONS.  BISECTION AND THE STURM          PEIGØØ4Ø
C       SEQUENCE PROPERTY ARE USED.  INITIAL UPPER AND LOWER BOUNDS     PEIGØØ5Ø
C       ARE DETERMINED FROM GERSCHGORIN ESTIMATES.                      PEIGØØ6Ø
C       VERSION DATE: FEBRUARY 1, 1977.                                 PEIGØØ7Ø
            DIMENSION D(1),E(1)                                         PEIGØØ8Ø
            COMMON /MACHEP/TOL                                          PEIGØØ9Ø
        E(N+1)=E(1)                                                     PEIGØ1ØØ
        NM1=N-1                                                         PEIGØ11Ø
        T=ABS(E(1))                                                     PEIGØ12Ø
        XU=D(1)                                                         PEIGØ13Ø
        XL=D(1)                                                         PEIGØ14Ø
        DO 5 I=1,N                                                      PEIGØ15Ø
            X=T                                                         PEIGØ16Ø
```

```
            T=ABS(E(I+1))                                                PEIGØ17Ø
            XU=AMAX1(XU,D(I)+(X+T))                                      PEIGØ18Ø
 5          XL=AMIN1(XL,D(I)-(X+T))                                      PEIGØ19Ø
        EPS=AMAX1(ABS(XU),ABS(XL))                                       PEIGØ2ØØ
        EPS1=1.E-2*EPS                                                   PEIGØ21Ø
        EPS=EPS*TOL                                                      PEIGØ22Ø
        XU=XU+EPS                                                        PEIGØ23Ø
        XL=XL-EPS                                                        PEIGØ24Ø
 1Ø     X=(XL+XU)/2.ØEØ                                                  PEIGØ25Ø
        T=EPS+4.ØEØ*TOL*(ABS(XL)+ABS(XU))                                PEIGØ26Ø
        IF((XU-XL).LE.T) GO TO 6Ø                                        PEIGØ27Ø
        KCOUNT=Ø                                                         PEIGØ28Ø
        ITEST=Ø                                                          PEIGØ29Ø
        Q=D(1)-X                                                         PEIGØ3ØØ
        T=E(1)                                                           PEIGØ31Ø
        C=D(N)-X                                                         PEIGØ32Ø
        IF(Q.LT.Ø.ØEØ) KCOUNT=1                                          PEIGØ33Ø
        DO 3Ø I=2,NM1                                                    PEIGØ34Ø
            Q1=Q                                                         PEIGØ35Ø
            T1=T                                                         PEIGØ36Ø
            IF(Q1.EQ.Ø.ØEØ) Q1=TOL*ABS(E(I))                            PEIGØ37Ø
            V=E(I)/Q1                                                    PEIGØ38Ø
            Q=D(I)-X-E(I)*V                                              PEIGØ39Ø
            T=T1*V                                                       PEIGØ4ØØ
            IF(Q.LT.Ø.ØEØ) KCOUNT=KCOUNT+1                               PEIGØ41Ø
            IF(ITEST.EQ.1) GO TO 25                                      PEIGØ42Ø
            IF(T1+C.EQ.C) GO TO 3Ø                                       PEIGØ43Ø
            IF(ABS(Q1).LE.EPS1) GO TO 2Ø                                 PEIGØ44Ø
            C=C-T1*T1/Q1                                                 PEIGØ45Ø
            GO TO 3Ø                                                     PEIGØ46Ø
 2Ø         C=C-T1*T1*(D(I)-X)/(Q1*Q)                                    PEIGØ47Ø
            ITEST=1                                                      PEIGØ48Ø
            GO TO 3Ø                                                     PEIGØ49Ø
 25         ITEST=Ø                                                      PEIGØ5ØØ
 3Ø         CONTINUE                                                     PEIGØ51Ø
        IF(Q.EQ.Ø.ØEØ) Q=TOL*ABS(E(N))                                   PEIGØ52Ø
        IF (ITEST) 35,35,4Ø                                              PEIGØ53Ø
 35     Q=C-(E(N)+T)*((E(N)+T)/Q)                                        PEIGØ54Ø
        GO TO 45                                                         PEIGØ55Ø
 4Ø     Q=C-E(N)*((E(N)+2.ØEØ*T)/Q)                                      PEIGØ56Ø
 45     IF(Q.LT.Ø.ØEØ) KCOUNT=KCOUNT+1                                   PEIGØ57Ø
        IF(KCOUNT-K) 5Ø,55,55                                            PEIGØ58Ø
 5Ø     XL=X                                                             PEIGØ59Ø
        GO TO 1Ø                                                         PEIGØ6ØØ
 55     XU=X                                                             PEIGØ61Ø
        GO TO 1Ø                                                         PEIGØ62Ø
 6Ø     EIGEN=X                                                          PEIGØ63Ø
        RETURN                                                           PEIGØ64Ø
        END                                                              PEIGØ65Ø


C       PROGRAM DRIVER                                                   DRVRØØ1Ø
C                                                                        DRVRØØ2Ø
C           THIS DRIVER GENERATES RANDOM PROBLEMS TO TEST SUBROUTINES    DRVRØØ3Ø
C       KPICK, GMA, AND GMAS.                                            DRVRØØ4Ø
C       THE USER MAY SPECIFY THE FOLLOWING PARAMETERS, WHICH APPEAR      DRVRØØ5Ø
C       AS THE FIRST FIVE EXECUTABLE STATEMENTS:                         DRVRØØ6Ø
C                                                                        DRVRØØ7Ø
C       ITMAX = THE NUMBER OF RANDOM PROBLEMS TO BE SOLVED.              DRVRØØ8Ø
C       NMAX = THE MAXIMUM VALUE OF N ALLOWED.                           DRVRØØ9Ø
C       MMAX = THE MAXIMUM VALUE OF M ALLOWED.                           DRVRØ1ØØ
C       IBDIM = THE ROW DIMENSION OF THE ARRAY B.                        DRVRØ11Ø
C       TOL = THE MACHINE EPSILON.                                       DRVRØ12Ø
C                                                                        DRVRØ13Ø
C       MINIMUM DIMENSIONS FOR THE ARRAYS ARE AS FOLLOWS:                DRVRØ14Ø
C       A: 5 * MAX( NMAX+1, MMAX+1 )                                     DRVRØ15Ø
C       B: MMAX + 2   X   NMAX + 2                                       DRVRØ16Ø
C       R: NMAX * INT( LOG2( NMAX ) ) + 3 * NMAX + 2                     DRVRØ17Ø
C       TN1, TN2, VN: NMAX + 1                                           DRVRØ18Ø
C       TM1, TM2, VM: MMAX + 1                                           DRVRØ19Ø
C                                                                        DRVRØ2ØØ
C       THE PROGRAM IS COMPLETE EXCEPT FOR THE TIMER, WHICH IS           DRVRØ21Ø
C       REFERRED TO AS THE INTEGER FORTRAN FUNCTION ITIMER(I).           DRVRØ22Ø
C                                                                        DRVRØ23Ø
C       VERSION DATE: MARCH 1, 1977.                                     DRVRØ24Ø
        COMMON /RAN1/INIT,IINIT,ISAVE                                    DRVRØ25Ø
```

```
      COMMON /MACHEP/TOL                                                DRVRØ26Ø
      DIMENSION A(16Ø),B(33,33),R(226)                                  DRVRØ27Ø
      DIMENSION TN1(32),TN2(32),VN(32),TM1(32),TM2(32),VM(32)           DRVRØ28Ø
      ITMAX=1ØØ                                                         DRVRØ29Ø
      NMAX=31                                                           DRVRØ3ØØ
      MMAX=31                                                           DRVRØ31Ø
      IBDIM=33                                                          DRVRØ32Ø
      TOL=2.ØEØ**(-23)                                                  DRVRØ33Ø
      INIT=1367                                                         DRVRØ34Ø
      IINIT=INIT                                                        DRVRØ35Ø
      D1=-ALOG1Ø(TOL)-1.ØEØ                                             DRVRØ36Ø
      DO 2Ø I=1,ITMAX                                                   DRVRØ37Ø
          N=INTRAN(3,NMAX)                                              DRVRØ38Ø
          M=INTRAN(3,MMAX)                                              DRVRØ39Ø
          NTYPE=INTRAN(1,4)                                             DRVRØ4ØØ
          MTYPE=INTRAN(1,4)                                             DRVRØ41Ø
          ITYPE=Ø                                                       DRVRØ42Ø
          JTYPE=Ø                                                       DRVRØ43Ø
          IF(NTYPE.EQ.2) ITYPE=INTRAN(1,9)                              DRVRØ44Ø
          IF(MTYPE.EQ.2) JTYPE=INTRAN(1,9)                              DRVRØ45Ø
          IDEF=INTRAN(Ø,3)                                              DRVRØ46Ø
          DEMAND=RAN(1.ØEØ,D1)                                          DRVRØ47Ø
          CALL TEVAL(N,NTYPE,ITYPE,TN1,TN2,IDEF)                        DRVRØ48Ø
          CALL TEVAL(M,MTYPE,JTYPE,TM1,TM2,IDEF)                        DRVRØ49Ø
          ITK=ITIMER(Ø)                                                 DRVRØ5ØØ
          CALL KPICK(Ø,N,NTYPE,TN1,TN2,M,MTYPE,TM1,TM2,                 DRVRØ51Ø
     1         DEMAND,K,COND,EMARCH,DIGITS,IFLAG)                       DRVRØ52Ø
          ITK=ITIMER(Ø)-ITK                                             DRVRØ53Ø
          IF(IDEF.GT.1) GO TO 5                                         DRVRØ54Ø
          ITR=ITIMER(Ø)                                                 DRVRØ55Ø
          CALL GMA(Ø,N,NTYPE,TN1,TN2,M,MTYPE,TM1,TM2,IBDIM,B,           DRVRØ56Ø
     1         K,R,A)                                                   DRVRØ57Ø
          ITR=ITIMER(Ø)-ITR                                             DRVRØ58Ø
          CALL RHS(N,NTYPE,TN1,TN2,M,MTYPE,TM1,TM2,IBDIM,B,             DRVRØ59Ø
     1         IDEF,VN,VM,COEFF)                                        DRVRØ6ØØ
          ITS=ITIMER(Ø)                                                 DRVRØ61Ø
          CALL GMA(1,N,NTYPE,TN1,TN2,M,MTYPE,TM1,TM2,IBDIM,B,           DRVRØ62Ø
     1         K,R,A)                                                   DRVRØ63Ø
          ITS=ITIMER(Ø)-ITS                                             DRVRØ64Ø
          GO TO 1Ø                                                      DRVRØ65Ø
     5    ITR=ITIMER(Ø)                                                 DRVRØ66Ø
          CALL GMAS(Ø,N,NTYPE,TN1,TN2,M,MTYPE,TM1,TM2,IBDIM,B,          DRVRØ67Ø
     1         VN,VM,K,R,A)                                             DRVRØ68Ø
          ITR=ITIMER(Ø)-ITR                                             DRVRØ69Ø
          CALL RHS(N,NTYPE,TN1,TN2,M,MTYPE,TM1,TM2,IBDIM,B,             DRVRØ7ØØ
     1         IDEF,VN,VM,COEFF)                                        DRVRØ71Ø
          ITS=ITIMER(Ø)                                                 DRVRØ72Ø
          CALL GMAS(1,N,NTYPE,TN1,TN2,M,MTYPE,TM1,TM2,IBDIM,B,          DRVRØ73Ø
     1         VN,VM,K,R,A)                                             DRVRØ74Ø
          ITS=ITIMER(Ø)-ITS                                             DRVRØ75Ø
    1Ø    ITG=ITR+ITS                                                   DRVRØ76Ø
          CALL NORM(ACTUAL,N,M,IBDIM,B,IDEF,VN,VM,COEFF)                DRVRØ77Ø
          IF(MOD(I,5Ø).EQ.1) WRITE(6,5Ø)                                DRVRØ78Ø
          WRITE(6,55) I,N,NTYPE,ITYPE,M,MTYPE,JTYPE,K,IFLAG,IDEF,       DRVRØ79Ø
     1         COND,EMARCH,DEMAND,DIGITS,ACTUAL,ITK,ITG,ITR,ITS         DRVRØ8ØØ
    2Ø    CONTINUE                                                      DRVRØ81Ø
    5Ø  FORMAT(1H1 / 46X,34HTHE GENERALIZED MARCHING ALGORITHM //       DRVRØ82Ø
      1 14X,18HPROBLEM PARAMETERS,17X,6HERRORS,13X,14HCORRECT DIGITS,   DRVRØ83Ø
      2 21X,7HTIMINGS / 4X,36HI   N NTYPE   M MTYPE     K    DEF,5X,   DRVRØ84Ø
      3 4ØHCOND       EMARCH   DEMAND DIGITS ACTUAL,4X,                 DRVRØ85Ø
      3 35HKPICK       GMA     ROOTS     SOLVE)                         DRVRØ86Ø
     55 FORMAT(2X,I3,1X,I3,2X,I1,2H (,I1,1H),1X,I3,2X,I1,2H (,I1,1H),   DRVRØ87Ø
      1 2X,I2,2H (,I1,1H),2X,I2,2X,E1Ø.3,2X,E1Ø.3,2X,F5.2,2X,F5.2,2X,   DRVRØ88Ø
      2 F5.2,2X,I8,2X,I8,2X,I8,2X,I8)                                   DRVRØ89Ø
      STOP                                                              DRVRØ9ØØ
      END                                                               DRVRØ91Ø
C
C
      FUNCTION RAN(XL,XU)                                               RANØØØ1Ø
C     FUNCTION RAN(XL,XU) GENERATES PSUEDO-RANDOM REAL NUMBERS          RANØØØ2Ø
C     ON THE CLOSED INTERVAL (XL,XU)                                    RANØØØ3Ø
C     VERSION DATE: FEBRUARY 1, 1977.                                   RANØØØ4Ø
          COMMON /RAN1/INIT,IINIT,ISAVE                                 RANØØØ5Ø
      INIT=MOD(3125*INIT,65536)                                         RANØØØ6Ø
```

```
      RAN=XL+(XU-XL)*FLOAT(INIT)*(2.ØEØ**(-16))                    RANØØØ7Ø
       RETURN                                                      RANØØØ8Ø
       END                                                         RANØØØ9Ø
C
C
      INTEGER FUNCTION INTRAN(IL,IU)                               IRANØØ1Ø
C     FUNCTION INTRAN(IL,IU) GENERATES PSUEDO-RANDOM INTEGERS      IRANØØ2Ø
C     ON THE CLOSED INTERVAL (IL,IU)                               IRANØØ3Ø
C     VERSION DATE: FEBRUARY 1, 1977.                              IRANØØ4Ø
          COMMON /RAN1/IINIT,INIT,ISAVE                            IRANØØ5Ø
      INIT=MOD(3125*INIT,65536)                                    IRANØØ6Ø
      Q=FLOAT(INIT*(IU-IL+1))*(2.ØEØ**(-16))                       IRANØØ7Ø
      INTRAN=IL+INT(Q)                                             IRANØØ8Ø
      RETURN                                                       IRANØØ9Ø
      END                                                          IRANØ1ØØ
C
C
      SUBROUTINE TEVAL(N,NTYPE,ITYPE,D,E,IDEF)                     TEVLØØ1Ø
C         SUBROUTINE TEVAL GENERATES A MATRIX OF ALLOWABLE TYPE    TEVLØØ2Ø
C     USING PSUEDO-RANDOM NUMBERS FOR MATRIX ELEMENTS.             TEVLØØ3Ø
C     VERSION DATE: FEBRUARY 1, 1977.                              TEVLØØ4Ø
          DIMENSION D(1),E(1)                                      TEVLØØ5Ø
      Q=1.ØEØ                                                      TEVLØØ6Ø
      IF((IDEF.EQ.1).OR.(IDEF.EQ.3)) Q=-Q                          TEVLØØ7Ø
      H=Q/FLOAT(N+1)                                               TEVLØØ8Ø
      IF(IDEF.GT.1) H=Ø.ØEØ                                        TEVLØØ9Ø
      H2=H*H*Q                                                     TEVLØ1ØØ
      XØ=Ø.ØEØ                                                     TEVLØ11Ø
      XL=Q*1.E-3                                                   TEVLØ12Ø
      XU=Q                                                         TEVLØ13Ø
      NP1=N+1                                                      TEVLØ14Ø
      IF(NTYPE.EQ.2.OR.NTYPE.EQ.4) GO TO 15                        TEVLØ15Ø
      DO 5 I=1,NP1                                                 TEVLØ16Ø
    5     E(I)=RAN(XL,XU)                                          TEVLØ17Ø
      IF(NTYPE.EQ.3) E(NP1)=E(1)                                   TEVLØ18Ø
      DO 1Ø I=1,N                                                  TEVLØ19Ø
   1Ø     D(I)=E(I)+E(I+1)+RAN(XØ,H2)                             TEVLØ2ØØ
      IF((IDEF.LE.1).OR.(NTYPE.EQ.3)) RETURN                       TEVLØ21Ø
      D(1)=E(2)                                                    TEVLØ22Ø
      D(N)=E(N)                                                    TEVLØ23Ø
      RETURN                                                       TEVLØ24Ø
   15 T1=RAN(XL,XU)                                                TEVLØ25Ø
      T2=2.ØEØ*T1+RAN(XØ,H2)                                       TEVLØ26Ø
      DO 2Ø I=1,NP1                                                TEVLØ27Ø
          D(I)=T2                                                  TEVLØ28Ø
   2Ø     E(I)=T1                                                  TEVLØ29Ø
      IF(NTYPE.EQ.4) RETURN                                        TEVLØ3ØØ
      S2=SQRT(2.ØEØ)*T1                                            TEVLØ31Ø
      IF(IDEF.GT.1) ITYPE=5                                        TEVLØ32Ø
      IF(ITYPE.LE.3) GO TO 3Ø                                      TEVLØ33Ø
      IF(ITYPE.GT.6) GO TO 25                                      TEVLØ34Ø
      L=INTRAN(Ø,1)                                                TEVLØ35Ø
      IF(L.EQ.Ø) E(2)=S2                                           TEVLØ36Ø
      IF(L.EQ.1) D(1)=T2-T1                                        TEVLØ37Ø
      GO TO 3Ø                                                     TEVLØ38Ø
   25 E(2)=S2                                                      TEVLØ39Ø
      D(1)=T2+RAN(XØ,H)                                            TEVLØ4ØØ
   3Ø I=ITYPE-(ITYPE/3)*3                                          TEVLØ41Ø
      IF(I.EQ.1) RETURN                                            TEVLØ42Ø
      IF(I.EQ.Ø) GO TO 4Ø                                          TEVLØ43Ø
      L=INTRAN(Ø,1)                                                TEVLØ44Ø
      IF(L.EQ.Ø) E(N)=S2                                           TEVLØ45Ø
      IF(L.EQ.1) D(N)=T2-T1                                        TEVLØ46Ø
      RETURN                                                       TEVLØ47Ø
   4Ø E(N)=S2                                                      TEVLØ48Ø
      D(N)=T2+RAN(XØ,H)                                            TEVLØ49Ø
      RETURN                                                       TEVLØ5ØØ
      END                                                          TEVLØ51Ø
C                                                                  TEVLØ52Ø
C
C
      SUBROUTINE RHS(N,NTYPE,TN1,TN2,M,MTYPE,TM1,TM2,IBDIM,B,      RHSØØØ1Ø
     1     IDEF,VN,VM,COEFF)                                       RHSØØØ2Ø
C         SUBROUTINE RHS GENERATES A RIGHT HAND SIDE FOR SUBROUTINE RHSØØØ3Ø
C     GMA USING PSUEDO-RANDOM NUMBERS.  ISAVE STORES THE CURRENT VALUERHSØØØ4Ø
```

```
C       OF INIT, SO THAT SUBROUTINE NORM CAN REGENERATE THE EXACT       RHSØØØ5Ø
C
C       SOLUTION.                                                       RHSØØØ6Ø
C       VERSION DATE: FEBRUARY 1, 1977.                                 RHSØØØ7Ø
            DIMENSION TN1(1),TN2(1),TM1(1),TM2(1),B(IBDIM,1),VN(1),VM(1)RHSØØØ8Ø
            COMMON /RAN1/INIT,IINIT,ISAVE                               RHSØØØ9Ø
            DOUBLE PRECISION S                                          RHSØØ1ØØ
        ISAVE=INIT                                                      RHSØØ11Ø
        XL=-1.ØEØ                                                       RHSØØ12Ø
        XU=1.ØEØ                                                        RHSØØ13Ø
        NP1=N+1                                                         RHSØØ14Ø
        MP1=M+1                                                         RHSØØ15Ø
        DO 5 I=2,MP1                                                    RHSØØ16Ø
        DO 5 J=2,NP1                                                    RHSØØ17Ø
    5       B(I,J)=RAN(XL,XU)                                           RHSØØ18Ø
        IF(MTYPE.GT.2) GO TO 15                                         RHSØØ19Ø
        TM2(1)=Ø.ØEØ                                                    RHSØØ2ØØ
        DO 1Ø I=2,NP1                                                   RHSØØ21Ø
            B(1,I)=Ø.ØEØ                                                RHSØØ22Ø
   1Ø       B(M+2,I)=Ø.ØEØ                                              RHSØØ23Ø
        GO TO 25                                                        RHSØØ24Ø
   15   DO 2Ø I=2,NP1                                                   RHSØØ25Ø
            B(1,I)=B(MP1,I)                                             RHSØØ26Ø
   2Ø       B(M+2,I)=B(2,I)                                             RHSØØ27Ø
   25   IF(NTYPE.GT.2) GO TO 35                                         RHSØØ28Ø
        TN2(1)=Ø.ØEØ                                                    RHSØØ29Ø
        DO 3Ø I=2,MP1                                                   RHSØØ3ØØ
            B(I,1)=Ø.ØEØ                                                RHSØØ31Ø
   3Ø       B(I,N+2)=Ø.ØEØ                                              RHSØØ32Ø
        GO TO 45                                                        RHSØØ33Ø
   35   DO 4Ø I=2,MP1                                                   RHSØØ34Ø
            B(I,1)=B(I,NP1)                                             RHSØØ35Ø
   4Ø       B(I,N+2)=B(I,2)                                             RHSØØ36Ø
   45   IF(IDEF.LE.1) GO TO 6Ø                                          RHSØØ37Ø
        COEFF=Ø.ØEØ                                                     RHSØØ38Ø
        DO 55 J=2,NP1                                                   RHSØØ39Ø
            T=Ø.ØEØ                                                     RHSØØ4ØØ
            DO 5Ø I=2,MP1                                               RHSØØ41Ø
   5Ø           T=T+VM(I-1)*B(I,J)                                      RHSØØ42Ø
   55       COEFF=COEFF+T*VN(J-1)                                       RHSØØ43Ø
   6Ø   TN2(NP1)=TN2(1)                                                 RHSØØ44Ø
        TM2(MP1)=TM2(1)                                                 RHSØØ45Ø
        DO 7Ø J=2,NP1                                                   RHSØØ46Ø
        DO 7Ø I=2,MP1                                                   RHSØØ47Ø
            S=(DBLE(TM1(I-1))+DBLE(TN1(J-1)))*DBLE(B(I,J))              RHSØØ48Ø
     1      -DBLE(TM2(I-1))*DBLE(B(I-1,J))                              RHSØØ49Ø
     2      -DBLE(TM2(I))*DBLE(B(I+1,J))                                RHSØØ5ØØ
     3      -DBLE(TN2(J-1))*DBLE(B(I,J-1))                              RHSØØ51Ø
     4      -DBLE(TN2(J))*DBLE(B(I,J+1))                                RHSØØ52Ø
   7Ø       B(I-1,J-1)=S                                                RHSØØ53Ø
        RETURN                                                          RHSØØ54Ø
        END                                                             RHSØØ55Ø
C
C
        SUBROUTINE NORM(DIGITS,N,M,IBDIM,B,IDEF,VN,VM,COEFF)            NORMØØ1Ø
C           SUBROUTINE NORM CALCULATES THE NUMBER OF SIGNIFICANT        NORMØØ2Ø
C       DIGITS IN THE COMPUTED SOLUTION, MEASURED IN THE 2-NORM.        NORMØØ3Ø
C       THE EXACT SOLUTION IS GENERATED AS REQUIRED USING THE VALUE     NORMØØ4Ø
C       OF ISAVE.                                                       NORMØØ5Ø
C       VERSION DATE: FEBRUARY 1, 1977.                                 NORMØØ6Ø
            DIMENSION B(IBDIM,1),VN(1),VM(1)                            NORMØØ7Ø
            COMMON /RAN1/INIT,IINIT,ISAVE                               NORMØØ8Ø
            DOUBLE PRECISION S                                          NORMØØ9Ø
        INIT=ISAVE                                                      NORMØ1ØØ
        XL=-1.ØEØ                                                       NORMØ11Ø
        XU=1.ØEØ                                                        NORMØ12Ø
        DIGITS=Ø.ØEØ                                                    NORMØ13Ø
        E1=Ø.ØEØ                                                        NORMØ14Ø
        E2=Ø.ØEØ                                                        NORMØ15Ø
        IF(IDEF.GT.1) GO TO 15                                          NORMØ16Ø
        DO 1Ø I=1,M                                                     NORMØ17Ø
        DO 1Ø J=1,N                                                     NORMØ18Ø
            U=RAN(XL,XU)                                                NORMØ19Ø
            S=DBLE(B(I,J))-DBLE(U)                                      NORMØ2ØØ
            E1=E1+S*S                                                   NORMØ21Ø
   1Ø       E2=E2+U*U                                                   NORMØ22Ø
        GO TO 25                                                        NORMØ23Ø
```

```
   15   DO 2Ø I=1,M                                                    NORMØ24Ø
            T=VM(I)*COEFF                                             NORMØ25Ø
        DO 2Ø J=1,N                                                   NORMØ26Ø
            U=RAN(XL,XU)-T*VN(J)                                      NORMØ27Ø
            S=DBLE(B(I,J))-DBLE(U)                                    NORMØ28Ø
            E1=E1+S*S                                                 NORMØ29Ø
   2Ø       E2=E2+U*U                                                 NORMØ3ØØ
   25   E1=E1/E2                                                      NORMØ31Ø
        IF(E1.LT.1.ØEØ) DIGITS=-ALOG1Ø(E1)/2.ØEØ                      NORMØ32Ø
        RETURN                                                        NORMØ33Ø
        END                                                           NORMØ34Ø
C
C
        SUBROUTINE KPICK(NPC,N,NTYPE,TN1,TN2,M,MTYPE,TM1,TM2,         KPIKØØ1Ø
      1     DEMAND,K,COND,EMARCH,DIGITS,IFLAG)                        KPIKØØ2Ø
C            SUBROUTINE KPICK IS DESIGNED TO AID THE USER IN SELECTING KPIKØØ3Ø
C       THE MARCHING PARAMETER K TO BE USED AS INPUT FOR SUBROUTINES  KPIKØØ4Ø
C       GMA AND GMAS.                                                 KPIKØØ5Ø
C                                                                     KPIKØØ6Ø
C       THE PARAMETER LIST:                                           KPIKØØ7Ø
C                                                                     KPIKØØ8Ø
C       NPC IS AN INTEGER.  IF NPC = Ø, SUBROUTINE KPICK DETERMINES K KPIKØØ9Ø
C           SUCH THAT THE SOLUTIONS COMPUTED USING SUBROUTINE GMA (GMAS)KPIKØ1ØØ
C           WILL HAVE APPROXIMATELY 'DEMAND' SIGNIFICANT DIGITS.      KPIKØ11Ø
C           IF NPC = 1, SUBROUTINE KPICK WILL TEST THE INPUT VALUE    KPIKØ12Ø
C           OF K.                                                     KPIKØ13Ø
C       N,NTYPE,TN1,TN2,M,MTYPE,TM1,TM2 ARE IDENTICAL TO THE          KPIKØ14Ø
C           CORRESPONDING PARAMETERS IN THE CALLING SEQUENCE FOR      KPIKØ15Ø
C           SUBROUTINE GMA (GMAS).                                    KPIKØ16Ø
C       DEMAND IS A REAL NUMBER, STATING THE NUMBER OF SIGNIFICANT    KPIKØ17Ø
C           DIGITS DESIRED IN THE COMPUTED SOLUTION.  IT MUST BE      KPIKØ18Ø
C           SPECIFIED ON INPUT IF NPC = Ø.                            KPIKØ19Ø
C       K IS AN INTEGER.  IF NPC = Ø, ON OUTPUT K IS EQUAL TO THE     KPIKØ2ØØ
C           MARCHING PARAMETER DETERMINED BY KPICK.  IF NPC = 1, THE  KPIKØ21Ø
C           USER MUST SPECIFY THE VALUE OF K TO BE TESTED AS AN       KPIKØ22Ø
C           INPUT PARAMETER.                                          KPIKØ23Ø
C       COND IS A REAL NUMBER, NORMALLY RETURNING THE CONDITION       KPIKØ24Ø
C           NUMBER OF THE LINEAR SYSTEM.                              KPIKØ25Ø
C       EMARCH IS A REAL NUMBER, NORMALLY RETURNING AN ESTIMATE OF    KPIKØ26Ø
C           THE ERROR DUE TO MARCHING FOR THE OUTPUT VALUE OF THE     KPIKØ27Ø
C           MARCHING PARAMETER K.                                     KPIKØ28Ø
C       DIGITS IS A REAL NUMBER, NORMALLY RETURNING A VALUE OF        KPIKØ29Ø
C           MAX( Ø.Ø, -ALOG1Ø( ( COND + EMARCH ) * TOL ), WHERE TOL   KPIKØ3ØØ
C           IS EQUAL TO THE MACHINE EPSILON.  THIS IS TYPICALLY THE   KPIKØ31Ø
C           NUMBER OF SIGNIFICANT DIGITS (IN THE 2 - NORM) ONE MAY    KPIKØ32Ø
C           EXPECT IN THE COMPUTED SOLUTION FOR THE GIVEN VALUE OF K. KPIKØ33Ø
C       IFLAG IS AN INTEGER DESCRIBING ERROR RETURNS.                 KPIKØ34Ø
C           IFLAG = Ø: NORMAL RETURN.                                 KPIKØ35Ø
C           IFLAG = 1: KPICK FAILED TO SUCCESSFULLY COMPUTE COND. THE KPIKØ36Ø
C               ALGORITHM USED BY KPICK ASSUMES THAT THE LINEAR       KPIKØ37Ø
C               SYSTEM IS EITHER POSITIVE OR NEGATIVE DEFINITE.       KPIKØ38Ø
C               IF THE LINEAR SYSTEM IS POSITIVE OR NEGATIVE          KPIKØ39Ø
C               SEMI-DEFINITE WITH A ZERO EIGENVALUE OF MULTIPLICITY  KPIKØ4ØØ
C               ONE, KPICK COMPUTES THE CONDITION NUMBER RELATIVE TO  KPIKØ41Ø
C               THE SUBSPACE ORTHOGONAL TO THE EIGENVECTOR ASSOCIATED KPIKØ42Ø
C               WITH THE ZERO EIGENVALUE.  OTHERWISE, KPICK RETURNS   KPIKØ43Ø
C               THE DEFAULT VALUES COND = EMARCH = DIGITS = Ø.Ø,      KPIKØ44Ø
C               AND K = 2 IF NPC = Ø.                                 KPIKØ45Ø
C           IFLAG = 2: KPICK HAS DETERMINED THAT THE MARCHING ERROR MAY KPIKØ46Ø
C               NOT SATISFY THE ASSUMPTIONS UNDERLYING THE ALGORITHM  KPIKØ47Ø
C               USED TO COMPUTE EMARCH, AND THUS THE COMPUTED VALUE   KPIKØ48Ø
C               OF DIGITS MAY BE IN DOUBT.  IF NPC = Ø, KPICK RETURNS KPIKØ49Ø
C               ESTIMATES FOR K = 2.  IF NPC = 1, KPICK RETURNS       KPIKØ5ØØ
C               ESTIMATES FOR THE INPUT VALUE OF K.                   KPIKØ51Ø
C           IFLAG = 3: CONDITIONS 1 AND 2 EXIST.                      KPIKØ52Ø
C           IFLAG = 4: THE REQUESTED DEMAND CANNOT BE SATISFIED BY ANY KPIKØ53Ø
C               VALUE OF K GREATER THAN OR EQUAL TO 2.  KPICK RETURNS KPIKØ54Ø
C               ESTIMATES FOR K = 2.   THIS ERROR RETURN CAN OCCUR ONLY KPIKØ55Ø
C               IF NPC = Ø.                                           KPIKØ56Ø
C           IFLAG = 5: CONDITIONS 1 AND 4 EXIST.                      KPIKØ57Ø
C           IFLAG = 6: CONDITIONS 2 AND 4 EXIST.                      KPIKØ58Ø
C           IFLAG = 7: CONDITIONS 1, 2, AND 4 EXIST.                  KPIKØ59Ø
C           IFLAG = 8: THE PARAMETERS N, NTYPE, TN1, AND/OR TN2 ARE   KPIKØ6ØØ
C               INCORRECTLY SPECIFIED.                                KPIKØ61Ø
C           IFLAG = 9: THE PARAMETERS M, MTYPE, TM1, AND/OR TM2 ARE   KPIKØ62Ø
C               INCORRECTLY SPECIFIED.                                KPIKØ63Ø
```

```
C           IFLAG = 1Ø: CONDITIONS 8 AND 9 EXIST.                          KPIKØ64Ø
C                                                                          KPIKØ65Ø
C           IF IFLAG = Ø, 2, 4, 6, THEN THE LINEAR SYSTEM IS SUITABLE      KPIKØ66Ø
C       FOR SUBROUTINE GMA, WITH THE QUALIFICATIONS NOTED ABOVE.           KPIKØ67Ø
C       IF IFLAG = 1, 3, 5, 7, WITH OTHER THAN DEFAULT VALUES FOR          KPIKØ68Ø
C       COND, EMARCH, AND DIGITS, THEN THE LINEAR SYSTEM IS SUITABLE       KPIKØ69Ø
C       FOR SUBROUTINE GMAS, WITH THE QUALIFICATIONS NOTED ABOVE.          KPIKØ7ØØ
C                                                                          KPIKØ71Ø
C           KPICK CONTAINS ONE LABELED COMMON BLOCK, /MACHEP/,             KPIKØ72Ø
C       CONTAINING ON VARIABLE, TOL.  TOL IS A MACHINE DEPENDENT           KPIKØ73Ø
C       CONSTANT EQUAL TO THE MACHINE EPSILON.  IT IS INITIALIZED IN       KPIKØ74Ø
C       KPICK IN THE FIRST EXECUTABLE STATEMENT.                           KPIKØ75Ø
C                                                                          KPIKØ76Ø
C       ADDRESS INQUIRIES TO:                                              KPIKØ77Ø
C           RANDOLPH E. BANK                                               KPIKØ78Ø
C           DEPARTMENT OF MATHEMATICS                                      KPIKØ79Ø
C           THE UNIVERSITY OF CHICAGO                                      KPIKØ8ØØ
C           CHICAGO, ILLINOIS 6Ø637                                        KPIKØ81Ø
C                                                                          KPIKØ82Ø
C       VERSION DATE: FEBRUARY 1, 1977.                                    KPIKØ83Ø
            DIMENSION TN1(1),TN2(1),TM1(1),TM2(1)                          KPIKØ84Ø
            COMMON /MACHEP/TOL                                             KPIKØ85Ø
        TOL=2.ØEØ**(-23)                                                   KPIKØ86Ø
C       ESTABLISH DEFAULT VALUES                                           KPIKØ87Ø
        IFLAG=Ø                                                            KPIKØ88Ø
        IF(NPC.NE.1) K=2                                                   KPIKØ89Ø
        COND=Ø.ØEØ                                                         KPIKØ9ØØ
        EMARCH=Ø.ØEØ                                                       KPIKØ91Ø
        DIGITS=Ø.ØEØ                                                       KPIKØ92Ø
        NN=N                                                               KPIKØ93Ø
        IF(NTYPE.GE.3) NN=N-1                                              KPIKØ94Ø
C       CHECK MATRIX SPECIFICATIONS.                                       KPIKØ95Ø
        JFLAG=8                                                            KPIKØ96Ø
        CALL TCHECK(N,NTYPE,TN1,TN2,IFLAG,JFLAG)                           KPIKØ97Ø
        JFLAG=9+IFLAG/8                                                    KPIKØ98Ø
        CALL TCHECK(M,MTYPE,TM1,TM2,IFLAG,JFLAG)                           KPIKØ99Ø
        IF(IFLAG.NE.Ø) RETURN                                              KPIK1ØØØ
C       DETERMINE IF THE SYSTEM IS POSITIVE OR NEGATIVE DEFINITE.          KPIK1Ø1Ø
        NS=2                                                               KPIK1Ø2Ø
        MS=2                                                               KPIK1Ø3Ø
        CALL EIGEN(N,NTYPE,TN1,TN2,1,TNMIN)                                KPIK1Ø4Ø
        CALL EIGEN(N,NTYPE,TN1,TN2,N,TNMAX)                                KPIK1Ø5Ø
        CALL EIGEN(M,MTYPE,TM1,TM2,1,TMMIN)                                KPIK1Ø6Ø
        CALL EIGEN(M,MTYPE,TM1,TM2,M,TMMAX)                                KPIK1Ø7Ø
        IF(ABS(TMMAX+TNMAX).GT.ABS(TMMIN+TNMAX)) GO TO 5                   KPIK1Ø8Ø
        MS=M-1                                                             KPIK1Ø9Ø
        T1=TMMAX                                                           KPIK11ØØ
        TMMAX=TMMIN                                                        KPIK111Ø
        TMMIN=T1                                                           KPIK112Ø
    5   IF(ABS(TMMAX+TNMAX).GT.ABS(TMMAX+TNMIN)) GO TO 1Ø                  KPIK113Ø
        NS=N-1                                                             KPIK114Ø
        T1=TNMAX                                                           KPIK115Ø
        TNMAX=TNMIN                                                        KPIK116Ø
        TNMIN=T1                                                           KPIK117Ø
   1Ø   T1=TMMAX+TNMAX                                                     KPIK118Ø
        T2=TMMIN+TNMIN                                                     KPIK119Ø
        IF(T2/T1.GT.TOL) GO TO 15                                          KPIK12ØØ
C       IF THE MATRIX IS NOT POSITIVE OR NEGATIVE DEFINITE,                KPIK121Ø
C       DETERMINE IF IT IS POSITIVE OR NEGATIVE SEMI-DEFINITE              KPIK122Ø
C       WITH A ZERO EIGENVALUE OF MULTIPLICITY ONE.                        KPIK123Ø
        IFLAG=1                                                            KPIK124Ø
        IF(ABS(T2).GT.ABS(T1)*TOL) RETURN                                  KPIK125Ø
        CALL EIGEN(N,NTYPE,TN1,TN2,NS,TNSNG)                               KPIK126Ø
        CALL EIGEN(M,MTYPE,TM1,TM2,MS,TMSNG)                               KPIK127Ø
        T2=TMMIN+TNSNG                                                     KPIK128Ø
        T3=TMSNG+TNMIN                                                     KPIK129Ø
        IF(ABS(T2).GT.ABS(T3)) T2=T3                                       KPIK13ØØ
        IF(T2/T1.LE.TOL) RETURN                                            KPIK131Ø
C       DETERMINE THE CONDITION NUMBER OF THE LINEAR SYSTEM.               KPIK132Ø
C       THE VALUE GF + SQRT( GF * GF - 1. ) IS AN ESTIMATE OF THE          KPIK133Ø
C       EXPONENTIAL GROWTH PER MARCHING STEP.                              KPIK134Ø
   15   COND=T1/T2                                                         KPIK135Ø
        GF=(2.ØEØ*TMMAX+TNMAX+TNMIN)/(TNMAX-TNMIN)                         KPIK136Ø
        BIG=COND/TOL                                                       KPIK137Ø
        IF(NPC.EQ.1) GO TO 5Ø                                              KPIK138Ø
C       ESTIMATE AN UPPER BOUND FOR K, KMAX, BASED ON AN EXPONENTIAL       KPIK139Ø
```

```
C       GROWTH OF 2.5 PER MARCHING STEP.                                        KPIK14ØØ
C       IF KEST SATISFIES DEMAND, CHECK THE VALUE OF GF.  IF THE GROWTH KPIK141Ø
C       IS LESS THAN 2.5 SET IFLAG = 2.  OTHERWISE ACCEPT                       KPIK142Ø
C       KEST AS A SUITABLE VALUE FOR K.                                         KPIK143Ø
        IF(DEMAND.LE.Ø.ØEØ) GO TO 45                                            KPIK144Ø
        TARGET=(1Ø.ØEØ**(-DEMAND))/TOL-COND                                     KPIK145Ø
        IF(TARGET.LE.15.625EØ) GO TO 45                                         KPIK146Ø
        KEST=ALOG1Ø(TARGET)/ALOG1Ø(2.5EØ)                                       KPIK147Ø
        IF(IFLAG.EQ.1) KEST=MINØ(KEST,N)                                        KPIK148Ø
        CALL ERROR(EMARCH,BIG,TMMAX,NN,TN1,TN2,KEST,ND)                         KPIK149Ø
        IF(EMARCH.GT.TARGET) GO TO 2Ø                                           KPIK15ØØ
        IF(GF.LE.1.45EØ) GO TO 5Ø                                               KPIK151Ø
        GO TO 4Ø                                                                KPIK152Ø
   2Ø   KMAX=NN/(ND+1)+1                                                        KPIK153Ø
C       CHECK IF KMIN = 2 IS A LOWER BOUND FOR K.  IF NOT SET IFLAG = 4.KPIK154Ø
        KMIN=2                                                                  KPIK155Ø
        CALL ERROR(EMARCH,BIG,TMMAX,NN,TN1,TN2,KMIN,ND)                         KPIK156Ø
        IF(TARGET.LT.EMARCH) GO TO 45                                           KPIK157Ø
C       FIND K USING THE METHOD OF BISECTION.                                   KPIK158Ø
   25   KEST=(KMAX+KMIN)/2                                                      KPIK159Ø
        CALL ERROR(EMARCH,BIG,TMMAX,NN,TN1,TN2,KEST,ND)                         KPIK16ØØ
        IF(KEST.EQ.KMIN) GO TO 4Ø                                               KPIK161Ø
        IF(TARGET-EMARCH) 3Ø,4Ø,35                                              KPIK162Ø
   3Ø   KMAX=NN/(ND+1)+1                                                        KPIK163Ø
        GO TO 25                                                                KPIK164Ø
   35   IF(ND.EQ.Ø) GO TO 4Ø                                                    KPIK165Ø
        KMIN=NN/ND                                                              KPIK166Ø
        GO TO 25                                                                KPIK167Ø
   4Ø   K=KEST                                                                  KPIK168Ø
        IF(K.NE.2) K=NN/(ND+1)+1                                                KPIK169Ø
        T1=(COND+EMARCH)*TOL                                                    KPIK17ØØ
        IF(T1.LT.1.ØEØ) DIGITS=-ALOG1Ø(T1)                                      KPIK171Ø
        RETURN                                                                  KPIK172Ø
C       IF NPC = 1, TEST THE INPUT VALUE OF K.  IF NPC = Ø,                     KPIK173Ø
C       TEST THE VALUE K = 2 FOR ERROR RETURNS.                                 KPIK174Ø
   45   IFLAG=IFLAG+4                                                           KPIK175Ø
   5Ø   KEST=K                                                                  KPIK176Ø
        IF(IFLAG.EQ.1) KEST=MINØ(KEST,N)                                        KPIK177Ø
        CALL ERROR(EMARCH,BIG,TMMAX,NN,TN1,TN2,KEST,ND)                         KPIK178Ø
        IF(GF.LE.1.45EØ) IFLAG=IFLAG+2                                          KPIK179Ø
        T1=(COND+EMARCH)*TOL                                                    KPIK18ØØ
        IF(T1.LT.1.ØEØ) DIGITS=-ALOG1Ø(T1)                                      KPIK181Ø
        RETURN                                                                  KPIK182Ø
        END                                                                     KPIK183Ø
C
C
        SUBROUTINE TCHECK(N,NTYPE,D,E,IFLAG,JFLAG)                              TCHKØØ1Ø
C           SUBROUTINE TCHECK DETERMINES IF THE INPUT DATA HAS BEEN             TCHKØØ2Ø
C       CORRECTLY SPECIFIED BY THE USER.                                        TCHKØØ3Ø
C       VERSION DATE: FEBRUARY 1, 1977.                                         TCHKØØ4Ø
            DIMENSION D(1),E(1)                                                 TCHKØØ5Ø
            COMMON /MACHEP/TOL                                                  TCHKØØ6Ø
        IF(N.LT.3) GO TO 5Ø                                                     TCHKØØ7Ø
        IF(NTYPE.LT.1) GO TO 5Ø                                                 TCHKØØ8Ø
        IF(NTYPE.GT.4) GO TO 5Ø                                                 TCHKØØ9Ø
        GO TO (5,25,1Ø,3Ø),NTYPE                                                TCHKØ1ØØ
    5   I1=2                                                                    TCHKØ11Ø
        GO TO 15                                                                TCHKØ12Ø
   1Ø   I1=1                                                                    TCHKØ13Ø
   15   DO 2Ø I=I1,N                                                            TCHKØ14Ø
            IF(E(I).EQ.Ø.ØEØ) GO TO 5Ø                                          TCHKØ15Ø
   2Ø   CONTINUE                                                               TCHKØ16Ø
        RETURN                                                                  TCHKØ17Ø
   25   IF(N.EQ.3) GO TO 5                                                      TCHKØ18Ø
        I1=3                                                                    TCHKØ19Ø
        I2=N-1                                                                  TCHKØ2ØØ
        GO TO 35                                                                TCHKØ21Ø
   3Ø   I1=1                                                                    TCHKØ22Ø
        I2=N                                                                    TCHKØ23Ø
   35   D1=D(2)                                                                 TCHKØ24Ø
        D2=ABS(D1)*1Ø.ØEØ*TOL                                                   TCHKØ25Ø
        IF(D1.EQ.Ø.ØEØ) D2=1Ø.ØEØ*TOL                                           TCHKØ26Ø
        E1=E(3)                                                                 TCHKØ27Ø
        E2=ABS(E1)*1Ø.ØEØ*TOL                                                   TCHKØ28Ø
        IF(E1.EQ.Ø.ØEØ) GO TO 5Ø                                                TCHKØ29Ø
        DO 4Ø I=I1,I2                                                           TCHKØ3ØØ
```

```
          IF(ABS(D1-D(I)).GT.D2) GO TO 5Ø                    TCHKØ31Ø
          IF(ABS(E1-E(I)).GT.E2) GO TO 5Ø                    TCHKØ32Ø
4Ø        CONTINUE                                           TCHKØ33Ø
          IF(E(2).EQ.Ø.ØEØ) GO TO 5Ø                         TCHKØ34Ø
          IF(E(N).EQ.Ø.ØEØ) GO TO 5Ø                         TCHKØ35Ø
          RETURN                                             TCHKØ36Ø
5Ø        IFLAG=JFLAG                                        TCHKØ37Ø
          RETURN                                             TCHKØ38Ø
          END                                                TCHKØ39Ø
```

ALGORITHM 528
Framework for a Portable Library [Z]

P. A. FOX, A. D. HALL, and N. L. SCHRYER
Bell Laboratories

Key Words and Phrases: portability, mathematical libraries, error handling, storage management, memory allocation, machine dependencies
CR Categories: 4.4, 5.1
Language: Fortran

DESCRIPTION

The three program packages presented here provide a framework for a portable Fortran subroutine library. They were developed for the Bell Laboratories library PORT [1]. The packages are: machine-dependent constants, automatic error handling, and dynamic storage allocation using a stack. There are interdependencies among the packages in the sense that the error handling is used by both the others, and it in turn uses machine-dependent constants provided by the first. However, care is taken to avoid any actual or apparent recursion.

Two non-ANSI Standard Fortran assumptions are made in the algorithm. The first is that there is no runtime subscript range checking; the second is that variables (local to a subprogram) initialized by DATA statements, and then changed within the subprogram, keep their values from one invocation of the subprogram to the next. Appendix A of [1] discusses nonstandard usage in more detail.

REFERENCES

1. FOX, P.A., HALL, A.D., AND SCHRYER, N.L. The PORT mathematical subroutine library. *ACM Trans. Math. Software 4,* 2 (June 1978), 104-126.

Machine-Dependent Constants

The first package contains three Fortran function subprograms which can be invoked to determine basic machine or operating system dependent constants. Values are provided in commented DATA statements for the Burroughs 5700/6700/7700, the CDC 6000/7000 series, the Data General Eclipse, DEC PDP 10 (KA and KI processors), the DEC PDP 11, the Harris S220, the Honeywell 6000 series, the IBM 360/370 series, the SEL systems 85/86, the Univac 1100 series, the XEROX SIGMA 5/7/9; others can be added. When the library is moved to a new environment, only the appropriate DATA statements in these three subprograms need to be activated by removing the C's from column 1.

The three functions are: I1MACH, which delivers integer constants, R1MACH,

Received 21 July 1976 and 9 May 1977.

Authors' address: Bell Laboratories, 600 Mountain Ave., Murray Hill, NJ 07974.

which delivers single-precision floating-point (REAL) constants, and D1MACH, which delivers double-precision floating-point constants. These functions have a single integer argument indicating the particular constant desired. For example, R1MACH(2) is the largest single-precision floating-point number on the host machine, so the statement

XMAX = R1MACH(2)

sets XMAX to this largest number.

The machine-dependent values provided are defined in the listing given here. (Please note that the systems for which the machine constants are given are printed here in alphabetical order but that their corresponding sequence numbers are not in numerical order.) Details of constant specification and usage are described in [1].

ALGORITHM

[Only the PORT utility programs are included here. A Users' Guide and test programs are available upon request from the ACM Algorithms Distribution Service.]

```
      INTEGER FUNCTION I1MACH(I)                                        MCHI0000
C                                                                       MCHI0020
C  I/O UNIT NUMBERS.                                                    MCHI0040
C                                                                       MCHI0060
C    I1MACH( 1) = THE STANDARD INPUT UNIT.                              MCHI0080
C                                                                       MCHI0100
C    I1MACH( 2) = THE STANDARD OUTPUT UNIT.                             MCHI0120
C                                                                       MCHI0140
C    I1MACH( 3) = THE STANDARD PUNCH UNIT.                              MCHI0160
C                                                                       MCHI0180
C    I1MACH( 4) = THE STANDARD ERROR MESSAGE UNIT.                      MCHI0200
C                                                                       MCHI0220
C  WORDS.                                                               MCHI0240
C                                                                       MCHI0260
C    I1MACH( 5) = THE NUMBER OF BITS PER INTEGER STORAGE UNIT.          MCHI0280
C                                                                       MCHI0300
C    I1MACH( 6) = THE NUMBER OF CHARACTERS PER INTEGER STORAGE UNIT.    MCHI0320
C                                                                       MCHI0340
C  INTEGERS.                                                            MCHI0360
C                                                                       MCHI0380
C    ASSUME INTEGERS ARE REPRESENTED IN THE S-DIGIT, BASE-A FORM        MCHI0400
C                                                                       MCHI0420
C                SIGN ( X(S-1)*A**(S-1) + ... + X(1)*A + X(0) )         MCHI0440
C                                                                       MCHI0460
C                WHERE 0 .LE. X(I) .LT. A FOR I=0,...,S-1.              MCHI0480
C                                                                       MCHI0500
C    I1MACH( 7) = A, THE BASE.                                          MCHI0520
C                                                                       MCHI0540
C    I1MACH( 8) = S, THE NUMBER OF BASE-A DIGITS.                       MCHI0560
C                                                                       MCHI0580
C    I1MACH( 9) = A**S - 1, THE LARGEST MAGNITUDE.                      MCHI0600
C                                                                       MCHI0620
C  FLOATING-POINT NUMBERS.                                              MCHI0640
C                                                                       MCHI0660
C    ASSUME FLOATING-POINT NUMBERS ARE REPRESENTED IN THE T-DIGIT,      MCHI0680
C    BASE-B FORM                                                        MCHI0700
C                                                                       MCHI0720
C                SIGN (B**E)*( (X(1)/B) + ... + (X(T)/B**T) )           MCHI0740
C                                                                       MCHI0760
C                WHERE 0 .LE. X(I) .LT. B FOR I=1,...,T,                MCHI0780
C                0 .LT. X(1), AND EMIN .LE. E .LE. EMAX.                MCHI0800
C                                                                       MCHI0820
C    I1MACH(10) = B, THE BASE.                                          MCHI0840
C                                                                       MCHI0860
C  SINGLE-PRECISION                                                     MCHI0880
C                                                                       MCHI0900
C    I1MACH(11) = T, THE NUMBER OF BASE-B DIGITS.                       MCHI0920
C                                                                       MCHI0940
```

```
C     I1MACH(12) = EMIN, THE SMALLEST EXPONENT E.                    MCHIØ96Ø
C                                                                    MCHIØ98Ø
C     I1MACH(13) = EMAX, THE LARGEST EXPONENT E.                     MCHI1ØØØ
C                                                                    MCHI1Ø2Ø
C  DOUBLE-PRECISION                                                  MCHI1Ø4Ø
C                                                                    MCHI1Ø6Ø
C     I1MACH(14) = T, THE NUMBER OF BASE-B DIGITS.                   MCHI1Ø8Ø
C                                                                    MCHI11ØØ
C     I1MACH(15) = EMIN, THE SMALLEST EXPONENT E.                    MCHI112Ø
C                                                                    MCHI114Ø
C     I1MACH(16) = EMAX, THE LARGEST EXPONENT E.                     MCHI116Ø
C                                                                    MCHI118Ø
C  TO ALTER THIS FUNCTION FOR A PARTICULAR ENVIRONMENT,              MCHI12ØØ
C  THE DESIRED SET OF DATA STATEMENTS SHOULD BE ACTIVATED BY         MCHI122Ø
C  REMOVING THE C FROM COLUMN 1.  ALSO, THE VALUES OF                MCHI124Ø
C  I1MACH(1) - I1MACH(4) SHOULD BE CHECKED FOR CONSISTENCY           MCHI126Ø
C  WITH THE LOCAL OPERATING SYSTEM.                                  MCHI128Ø
C                                                                    MCHI13ØØ
      INTEGER IMACH(16),OUTPUT                                       MCHI132Ø
C                                                                    MCHI134Ø
      EQUIVALENCE (IMACH(4),OUTPUT)                                  MCHI136Ø
C                                                                    MCHI138Ø
C     MACHINE CONSTANTS FOR THE BURROUGHS 1700 SYSTEM.               MCHI488Ø
C                                                                    MCHI49ØØ
C     DATA IMACH( 1) /    7 /                                        MCHI492Ø
C     DATA IMACH( 2) /    2 /                                        MCHI494Ø
C     DATA IMACH( 3) /    2 /                                        MCHI496Ø
C     DATA IMACH( 4) /    2 /                                        MCHI498Ø
C     DATA IMACH( 5) /   36 /                                        MCHI5ØØØ
C     DATA IMACH( 6) /    4 /                                        MCHI5Ø2Ø
C     DATA IMACH( 7) /    2 /                                        MCHI5Ø4Ø
C     DATA IMACH( 8) /   33 /                                        MCHI5Ø6Ø
C     DATA IMACH( 9) / Z1FFFFFFFF /                                  MCHI5Ø8Ø
C     DATA IMACH(1Ø) /    2 /                                        MCHI51ØØ
C     DATA IMACH(11) /   24 /                                        MCHI512Ø
C     DATA IMACH(12) / -256 /                                        MCHI514Ø
C     DATA IMACH(13) /  255 /                                        MCHI516Ø
C     DATA IMACH(14) /   6Ø /                                        MCHI518Ø
C     DATA IMACH(15) / -256 /                                        MCHI52ØØ
C     DATA IMACH(16) /  255 /                                        MCHI522Ø
C                                                                    MCHI524Ø
C     MACHINE CONSTANTS FOR THE BURROUGHS 57ØØ SYSTEM.               MCHI45ØØ
C                                                                    MCHI452Ø
C     DATA IMACH( 1) /   5 /                                         MCHI454Ø
C     DATA IMACH( 2) /   6 /                                         MCHI456Ø
C     DATA IMACH( 3) /   7 /                                         MCHI458Ø
C     DATA IMACH( 4) /   6 /                                         MCHI46ØØ
C     DATA IMACH( 5) /  48 /                                         MCHI462Ø
C     DATA IMACH( 6) /   6 /                                         MCHI464Ø
C     DATA IMACH( 7) /   2 /                                         MCHI466Ø
C     DATA IMACH( 8) /  39 /                                         MCHI468Ø
C     DATA IMACH( 9) / OØØØ7777777777777 /                           MCHI47ØØ
C     DATA IMACH(1Ø) /   8 /                                         MCHI472Ø
C     DATA IMACH(11) /  13 /                                         MCHI474Ø
C     DATA IMACH(12) / -5Ø /                                         MCHI476Ø
C     DATA IMACH(13) /  76 /                                         MCHI478Ø
C     DATA IMACH(14) /  26 /                                         MCHI48ØØ
C     DATA IMACH(15) / -5Ø /                                         MCHI482Ø
C     DATA IMACH(16) /  76 /                                         MCHI484Ø
C                                                                    MCHI486Ø
C     MACHINE CONSTANTS FOR THE BURROUGHS 67ØØ/77ØØ SYSTEMS.         MCHI412Ø
C                                                                    MCHI414Ø
C     DATA IMACH( 1) /   5 /                                         MCHI416Ø
C     DATA IMACH( 2) /   6 /                                         MCHI418Ø
C     DATA IMACH( 3) /   7 /                                         MCHI42ØØ
C     DATA IMACH( 4) /   6 /                                         MCHI422Ø
C     DATA IMACH( 5) /  48 /                                         MCHI424Ø
C     DATA IMACH( 6) /   6 /                                         MCHI426Ø
C     DATA IMACH( 7) /   2 /                                         MCHI428Ø
C     DATA IMACH( 8) /  39 /                                         MCHI43ØØ
C     DATA IMACH( 9) / OØØØ7777777777777 /                           MCHI432Ø
C     DATA IMACH(1Ø) /   8 /                                         MCHI434Ø
C     DATA IMACH(11) /  13 /                                         MCHI436Ø
C     DATA IMACH(12) / -5Ø /                                         MCHI438Ø
C     DATA IMACH(13) /  76 /                                         MCHI44ØØ
C     DATA IMACH(14) /  26 /                                         MCHI442Ø
```

```
C     DATA IMACH(15) / -32754 /                                    MCHI444Ø
C     DATA IMACH(16) /  3278Ø /                                    MCHI446Ø
C                                                                  MCHI448Ø
C     MACHINE CONSTANTS FOR THE CDC 6ØØØ/7ØØØ SERIES.              MCHI218Ø
C                                                                  MCHI22ØØ
C     DATA IMACH( 1) /    5 /                                      MCHI222Ø
C     DATA IMACH( 2) /    6 /                                      MCHI224Ø
C     DATA IMACH( 3) /    7 /                                      MCHI226Ø
C     DATA IMACH( 4) /    6 /                                      MCHI228Ø
C     DATA IMACH( 5) /   6Ø /                                      MCHI23ØØ
C     DATA IMACH( 6) /   1Ø /                                      MCHI232Ø
C     DATA IMACH( 7) /    2 /                                      MCHI234Ø
C     DATA IMACH( 8) /   48 /                                      MCHI236Ø
C     DATA IMACH( 9) / ØØØØ7777777777777777B /                     MCHI238Ø
C     DATA IMACH(1Ø) /    2 /                                      MCHI24ØØ
C     DATA IMACH(11) /   48 /                                      MCHI242Ø
C     DATA IMACH(12) / -974 /                                      MCHI244Ø
C     DATA IMACH(13) / 1Ø7Ø /                                      MCHI246Ø
C     DATA IMACH(14) /   96 /                                      MCHI248Ø
C     DATA IMACH(15) / -927 /                                      MCHI25ØØ
C     DATA IMACH(16) / 1Ø7Ø /                                      MCHI252Ø
C                                                                  MCHI254Ø
C     MACHINE CONSTANTS FOR THE CRAY 1
C
C     DATA IMACH( 1) /   1ØØ /
C     DATA IMACH( 2) /   1Ø1 /
C     DATA IMACH( 3) /   1Ø2 /
C     DATA IMACH( 4) /   1Ø1 /
C     DATA IMACH( 5) /    64 /
C     DATA IMACH( 6) /     8 /
C     DATA IMACH( 7) /     2 /
C     DATA IMACH( 8) /    63 /
C     DATA IMACH( 9) /  777777777777777777B /
C     DATA IMACH(1Ø) /     2 /
C     DATA IMACH(11) /    48 /
C     DATA IMACH(12) / -8192 /
C     DATA IMACH(13) /  8191 /
C     DATA IMACH(14) /    96 /
C     DATA IMACH(15) / -8192 /
C     DATA IMACH(16) /  8191 /
C
C     MACHINE CONSTANTS FOR THE DATA GENERAL ECLIPSE S/2ØØ         MCHI572Ø
C                                                                  MCHI574Ø
C     DATA IMACH( 1) /   11 /                                      MCHI576Ø
C     DATA IMACH( 2) /   12 /                                      MCHI578Ø
C     DATA IMACH( 3) /    8 /                                      MCHI58ØØ
C     DATA IMACH( 4) /   1Ø /                                      MCHI582Ø
C     DATA IMACH( 5) /   16 /                                      MCHI584Ø
C     DATA IMACH( 6) /    2 /                                      MCHI586Ø
C     DATA IMACH( 7) /    2 /                                      MCHI588Ø
C     DATA IMACH( 8) /   15 /                                      MCHI59ØØ
C     DATA IMACH( 9) /32767 /                                      MCHI592Ø
C     DATA IMACH(1Ø) /   16 /                                      MCHI594Ø
C     DATA IMACH(11) /    6 /                                      MCHI596Ø
C     DATA IMACH(12) /  -64 /                                      MCHI598Ø
C     DATA IMACH(13) /   63 /                                      MCHI6ØØØ
C     DATA IMACH(14) /   14 /                                      MCHI6Ø2Ø
C     DATA IMACH(15) /  -64 /                                      MCHI6Ø4Ø
C     DATA IMACH(16) /   63 /                                      MCHI6Ø6Ø
C                                                                  MCHI6Ø8Ø
C     MACHINE CONSTANTS FOR THE HARRIS 22Ø                         MCHI61ØØ
C                                                                  MCHI612Ø
C     DATA IMACH( 1) /       5 /                                   MCHI614Ø
C     DATA IMACH( 2) /       6 /                                   MCHI616Ø
C     DATA IMACH( 3) /       Ø /                                   MCHI618Ø
C     DATA IMACH( 4) /       6 /                                   MCHI62ØØ
C     DATA IMACH( 5) /      24 /                                   MCHI622Ø
C     DATA IMACH( 6) /       3 /                                   MCHI624Ø
C     DATA IMACH( 7) /       2 /                                   MCHI626Ø
C     DATA IMACH( 8) /      23 /                                   MCHI628Ø
C     DATA IMACH( 9) / 8388607 /                                   MCHI63ØØ
C     DATA IMACH(1Ø) /       2 /                                   MCHI632Ø
C     DATA IMACH(11) /      23 /                                   MCHI634Ø
C     DATA IMACH(12) /    -127 /                                   MCHI636Ø
C     DATA IMACH(13) /     127 /                                   MCHI638Ø
C     DATA IMACH(14) /      38 /                                   MCHI64ØØ
```

```
C     DATA IMACH(15) /    -127 /                                        MCHI642Ø
C     DATA IMACH(16) /     127 /                                        MCHI644Ø
C                                                                       MCHI646Ø
C     MACHINE CONSTANTS FOR THE HONEYWELL 6ØØ/6ØØØ SERIES.              MCHI14ØØ
C                                                                       MCHI142Ø
C     DATA IMACH( 1) /    5 /                                           MCHI144Ø
C     DATA IMACH( 2) /    6 /                                           MCHI146Ø
C     DATA IMACH( 3) /   43 /                                           MCHI148Ø
C     DATA IMACH( 4) /    6 /                                           MCHI15ØØ
C     DATA IMACH( 5) /   36 /                                           MCHI152Ø
C     DATA IMACH( 6) /    6 /                                           MCHI154Ø
C     DATA IMACH( 7) /    2 /                                           MCHI156Ø
C     DATA IMACH( 8) /   35 /                                           MCHI158Ø
C     DATA IMACH( 9) / 0377777777777 /                                  MCHI16ØØ
C     DATA IMACH(1Ø) /    2 /                                           MCHI162Ø
C     DATA IMACH(11) /   27 /                                           MCHI164Ø
C     DATA IMACH(12) / -127 /                                           MCHI166Ø
short course: intro to tso,part 2, thursday 16 march, 7-9 pm, room 354 m1 OPER
C     DATA IMACH(13) /  127 /                                           MCHI168Ø
C     DATA IMACH(14) /   63 /                                           MCHI17ØØ
C     DATA IMACH(15) / -127 /                                           MCHI172Ø
C     DATA IMACH(16) /  127 /                                           MCHI174Ø
C                                                                       MCHI176Ø
C     MACHINE CONSTANTS FOR THE IBM 36Ø/37Ø SERIES,                     MCHI178Ø
C     THE XEROX SIGMA 5/7/9 AND THE SEL SYSTEMS 85/86.                  MCHI18ØØ
C                                                                       MCHI182Ø
C     DATA IMACH( 1) /    5 /                                           MCHI184Ø
C     DATA IMACH( 2) /    6 /                                           MCHI186Ø
C     DATA IMACH( 3) /    7 /                                           MCHI188Ø
C     DATA IMACH( 4) /    6 /                                           MCHI19ØØ
C     DATA IMACH( 5) /   32 /                                           MCHI192Ø
C     DATA IMACH( 6) /    4 /                                           MCHI194Ø
C     DATA IMACH( 7) /    2 /                                           MCHI196Ø
C     DATA IMACH( 8) /   31 /                                           MCHI198Ø
C     DATA IMACH( 9) / Z7FFFFFFF /                                      MCHI2ØØØ
C     DATA IMACH(1Ø) /   16 /                                           MCHI2Ø2Ø
C     DATA IMACH(11) /    6 /                                           MCHI2Ø4Ø
C     DATA IMACH(12) / -64 /                                            MCHI2Ø6Ø
C     DATA IMACH(13) /   63 /                                           MCHI2Ø8Ø
C     DATA IMACH(14) /   14 /                                           MCHI21ØØ
C     DATA IMACH(15) / -64 /                                            MCHI212Ø
C     DATA IMACH(16) /   63 /                                           MCHI214Ø
C                                                                       MCHI216Ø
C     MACHINE CONSTANTS FOR THE PDP-1Ø (KA PROCESSOR).                  MCHI256Ø
C                                                                       MCHI258Ø
C     DATA IMACH( 1) /    5 /                                           MCHI26ØØ
C     DATA IMACH( 2) /    6 /                                           MCHI262Ø
C     DATA IMACH( 3) /    5 /                                           MCHI264Ø
C     DATA IMACH( 4) /    6 /                                           MCHI266Ø
C     DATA IMACH( 5) /   36 /                                           MCHI268Ø
C     DATA IMACH( 6) /    5 /                                           MCHI27ØØ
C     DATA IMACH( 7) /    2 /                                           MCHI272Ø
C     DATA IMACH( 8) /   35 /                                           MCHI274Ø
C     DATA IMACH( 9) / "377777777777 /                                  MCHI276Ø
C     DATA IMACH(1Ø) /    2 /                                           MCHI278Ø
C     DATA IMACH(11) /   27 /                                           MCHI28ØØ
C     DATA IMACH(12) / -128 /                                           MCHI282Ø
C     DATA IMACH(13) /  127 /                                           MCHI284Ø
C     DATA IMACH(14) /   54 /                                           MCHI286Ø
C     DATA IMACH(15) / -1Ø1 /                                           MCHI288Ø
C     DATA IMACH(16) /  127 /                                           MCHI29ØØ
C                                                                       MCHI292Ø
C     MACHINE CONSTANTS FOR THE PDP-1Ø (KI PROCESSOR).                  MCHI294Ø
C                                                                       MCHI296Ø
C     DATA IMACH( 1) /    5 /                                           MCHI298Ø
C     DATA IMACH( 2) /    6 /                                           MCHI3ØØØ
C     DATA IMACH( 3) /    5 /                                           MCHI3Ø2Ø
C     DATA IMACH( 4) /    6 /                                           MCHI3Ø4Ø
C     DATA IMACH( 5) /   36 /                                           MCHI3Ø6Ø
C     DATA IMACH( 6) /    5 /                                           MCHI3Ø8Ø
C     DATA IMACH( 7) /    2 /                                           MCHI31ØØ
C     DATA IMACH( 8) /   35 /                                           MCHI312Ø
C     DATA IMACH( 9) / "377777777777 /                                  MCHI314Ø
C     DATA IMACH(1Ø) /    2 /                                           MCHI316Ø
C     DATA IMACH(11) /   27 /                                           MCHI318Ø
C     DATA IMACH(12) / -128 /                                           MCHI32ØØ
```

```
C     DATA IMACH(13) /  127 /                                    MCHI322Ø
C     DATA IMACH(14) /   62 /                                    MCHI324Ø
C     DATA IMACH(15) / -128 /                                    MCHI326Ø
C     DATA IMACH(16) /  127 /                                    MCHI328Ø
C                                                                MCHI33ØØ
C     MACHINE CONSTANTS FOR PDP-11 FORTRAN'S SUPPORTING          MCHI332Ø
C     32-BIT INTEGER ARITHMETIC.                                 MCHI334Ø
C                                                                MCHI336Ø
C     DATA IMACH( 1) /    5 /                                    MCHI338Ø
C     DATA IMACH( 2) /    6 /                                    MCHI34ØØ
C     DATA IMACH( 3) /    5 /                                    MCHI342Ø
C     DATA IMACH( 4) /    6 /                                    MCHI344Ø
C     DATA IMACH( 5) /   32 /                                    MCHI346Ø
C     DATA IMACH( 6) /    4 /                                    MCHI348Ø
C     DATA IMACH( 7) /    2 /                                    MCHI35ØØ
C     DATA IMACH( 8) /   31 /                                    MCHI352Ø
C     DATA IMACH( 9) / 2147483647 /                              MCHI354Ø
C     DATA IMACH(1Ø) /    2 /                                    MCHI356Ø
C     DATA IMACH(11) /   24 /                                    MCHI358Ø
C     DATA IMACH(12) / -127 /                                    MCHI36ØØ
C     DATA IMACH(13) /  127 /                                    MCHI362Ø
C     DATA IMACH(14) /   56 /                                    MCHI364Ø
C     DATA IMACH(15) / -127 /                                    MCHI366Ø
C     DATA IMACH(16) /  127 /                                    MCHI368Ø
C                                                                MCHI37ØØ
C     MACHINE CONSTANTS FOR PDP-11 FORTRAN'S SUPPORTING          MCHI372Ø
C     16-BIT INTEGER ARITHMETIC.                                 MCHI374Ø
C                                                                MCHI376Ø
C     DATA IMACH( 1) /    5 /                                    MCHI378Ø
C     DATA IMACH( 2) /    6 /                                    MCHI38ØØ
C     DATA IMACH( 3) /    5 /                                    MCHI382Ø
C     DATA IMACH( 4) /    6 /                                    MCHI384Ø
C     DATA IMACH( 5) /   16 /                                    MCHI386Ø
C     DATA IMACH( 6) /    2 /                                    MCHI388Ø
C     DATA IMACH( 7) /    2 /                                    MCHI39ØØ
C     DATA IMACH( 8) /   15 /                                    MCHI392Ø
C     DATA IMACH( 9) / 32767 /                                   MCHI394Ø
C     DATA IMACH(1Ø) /    2 /                                    MCHI396Ø
C     DATA IMACH(11) /   24 /                                    MCHI398Ø
C     DATA IMACH(12) / -127 /                                    MCHI4ØØØ
C     DATA IMACH(13) /  127 /                                    MCHI4Ø2Ø
C     DATA IMACH(14) /   56 /                                    MCHI4Ø4Ø
C     DATA IMACH(15) / -127 /                                    MCHI4Ø6Ø
C     DATA IMACH(16) /  127 /                                    MCHI4Ø8Ø
C                                                                MCHI41ØØ
C     MACHINE CONSTANTS FOR THE UNIVAC 11ØØ SERIES.              MCHI526Ø
C                                                                MCHI528Ø
C     NOTE THAT THE PUNCH UNIT, I1MACH(3), HAS BEEN SET TO 7     MCHI53ØØ
C     WHICH IS APPROPRIATE FOR THE UNIVAC-FOR SYSTEM.            MCHI532Ø
C     IF YOU HAVE THE UNIVAC-FTN SYSTEM, SET IT TO 1.            MCHI534Ø
C                                                                MCHI536Ø
C     DATA IMACH( 1) /    5 /                                    MCHI538Ø
C     DATA IMACH( 2) /    6 /                                    MCHI54ØØ
C     DATA IMACH( 3) /    7 /                                    MCHI542Ø
C     DATA IMACH( 4) /    6 /                                    MCHI544Ø
C     DATA IMACH( 5) /   36 /                                    MCHI546Ø
C     DATA IMACH( 6) /    6 /                                    MCHI548Ø
C     DATA IMACH( 7) /    2 /                                    MCHI55ØØ
C     DATA IMACH( 8) /   35 /                                    MCHI552Ø
C     DATA IMACH( 9) / 0377777777777 /                           MCHI554Ø
C     DATA IMACH(1Ø) /    2 /                                    MCHI556Ø
C     DATA IMACH(11) /   27 /                                    MCHI558Ø
C     DATA IMACH(12) / -128 /                                    MCHI56ØØ
C     DATA IMACH(13) /  127 /                                    MCHI562Ø
C     DATA IMACH(14) /   6Ø /                                    MCHI564Ø
C     DATA IMACH(15) /-1Ø24 /                                    MCHI566Ø
C     DATA IMACH(16) / 1Ø23 /                                    MCHI568Ø
C                                                                MCHI57ØØ
      IF (I .LT. 1  .OR.  I .GT. 16) GO TO 1Ø                    MCHI648Ø
C                                                                MCHI65ØØ
      I1MACH=IMACH(I)                                            MCHI652Ø
      RETURN                                                     MCHI654Ø
C                                                                MCHI656Ø
 1Ø   WRITE(OUTPUT,9ØØØ)                                         MCHI658Ø
 9ØØØ FORMAT(39H1ERROR    1 IN I1MACH - I OUT OF BOUNDS)        MCHI66ØØ
C                                                                MCHI662Ø
```

```
      CALL FDUMP                                                        MCHI664Ø
C                                                                       MCHI666Ø
      STOP                                                              MCHI668Ø
C                                                                       MCHI67ØØ
      END                                                               MCHI672Ø
```

The function for REAL floating-point numbers is

```
      REAL FUNCTION R1MACH(I)                                           MCHRØØØØ
C                                                                       MCHRØØ2Ø
C  SINGLE-PRECISION MACHINE CONSTANTS                                   MCHRØØ4Ø
C                                                                       MCHRØØ6Ø
C  R1MACH(1) = B**(EMIN-1), THE SMALLEST POSITIVE MAGNITUDE.            MCHRØØ8Ø
C                                                                       MCHRØ1ØØ
C  R1MACH(2) = B**EMAX*(1 - B**(-T)), THE LARGEST MAGNITUDE.            MCHRØ12Ø
C                                                                       MCHRØ14Ø
C  R1MACH(3) = B**(-T), THE SMALLEST RELATIVE SPACING.                  MCHRØ16Ø
C                                                                       MCHRØ18Ø
C  R1MACH(4) = B**(1-T), THE LARGEST RELATIVE SPACING.                  MCHRØ2ØØ
C                                                                       MCHRØ22Ø
C  R1MACH(5) = LOG1Ø(B)                                                 MCHRØ24Ø
C                                                                       MCHRØ26Ø
C  TO ALTER THIS FUNCTION FOR A PARTICULAR ENVIRONMENT,                 MCHRØ28Ø
C  THE DESIRED SET OF DATA STATEMENTS SHOULD BE ACTIVATED BY            MCHRØ3ØØ
C  REMOVING THE C FROM COLUMN 1.                                        MCHRØ32Ø
C                                                                       MCHRØ34Ø
C  WHERE POSSIBLE, OCTAL OR HEXADECIMAL CONSTANTS HAVE BEEN USED        MCHRØ36Ø
C  TO SPECIFY THE CONSTANTS EXACTLY WHICH HAS IN SOME CASES             MCHRØ38Ø
C  REQUIRED THE USE OF EQUIVALENT INTEGER ARRAYS.                       MCHRØ4ØØ
C                                                                       MCHRØ42Ø
      INTEGER SMALL(2)                                                  MCHRØ44Ø
      INTEGER LARGE(2)                                                  MCHRØ46Ø
      INTEGER RIGHT(2)                                                  MCHRØ48Ø
      INTEGER DIVER(2)                                                  MCHRØ5ØØ
      INTEGER LOG1Ø(2)                                                  MCHRØ52Ø
C                                                                       MCHRØ54Ø
      REAL RMACH(5)                                                     MCHRØ56Ø
C                                                                       MCHRØ58Ø
      EQUIVALENCE (RMACH(1),SMALL(1))                                   MCHRØ6ØØ
      EQUIVALENCE (RMACH(2),LARGE(1))                                   MCHRØ62Ø
      EQUIVALENCE (RMACH(3),RIGHT(1))                                   MCHRØ64Ø
      EQUIVALENCE (RMACH(4),DIVER(1))                                   MCHRØ66Ø
      EQUIVALENCE (RMACH(5),LOG1Ø(1))                                   MCHRØ68Ø
C                                                                       MCHRØ7ØØ
C     MACHINE CONSTANTS FOR THE BURROUGHS 17ØØ SYSTEM.                  MCHR214Ø
C                                                                       MCHR216Ø
C     DATA RMACH(1) / Z4ØØ8ØØØØØ /                                      MCHR218Ø
C     DATA RMACH(2) / Z5FFFFFFFF /                                      MCHR22ØØ
C     DATA RMACH(3) / Z4E98ØØØØØ /                                      MCHR222Ø
C     DATA RMACH(4) / Z4EA8ØØØØØ /                                      MCHR224Ø
C     DATA RMACH(5) / Z5ØØE73ØE8 /                                      MCHR226Ø
C                                                                       MCHR228Ø
C     MACHINE CONSTANTS FOR THE BURROUGHS 57ØØ/67ØØ/77ØØ SYSTEMS.       MCHR198Ø
C                                                                       MCHR2ØØØ
C     DATA RMACH(1) / O1771ØØØØØØØØØØØØ /                              MCHR2Ø2Ø
C     DATA RMACH(2) / OØ777777777777777 /                              MCHR2Ø4Ø
C     DATA RMACH(3) / O1311ØØØØØØØØØØØ /                               MCHR2Ø6Ø
C     DATA RMACH(4) / O13Ø1ØØØØØØØØØØØ /                               MCHR2Ø8Ø
C     DATA RMACH(5) / O1157163Ø34761675 /                               MCHR21ØØ
C                                                                       MCHR212Ø
C     MACHINE CONSTANTS FOR THE CDC 6ØØØ/7ØØØ SERIES.                   MCHR1Ø6Ø
C                                                                       MCHR1Ø8Ø
C     DATA RMACH(1) / ØØØ14ØØØØØØØØØØØØØØØB /                          MCHR11ØØ
C     DATA RMACH(2) / 377677777777777777B /                             MCHR112Ø
C     DATA RMACH(3) / 16Ø4ØØØØØØØØØØØØØØØB /                           MCHR114Ø
C     DATA RMACH(4) / 1641ØØØØØØØØØØØØØØØB /                           MCHR116Ø
C     DATA RMACH(5) / 17164642Ø23241175721ØB /                          MCHR118Ø
C                                                                       MCHR12ØØ
C     MACHINE CONSTANTS FOR THE CRAY 1
C
C     DATA RMACH(1) / 2ØØØØ4ØØØØØØØØØØØØØØØB /
C     DATA RMACH(2) / 577777777777777777B /
C     DATA RMACH(3) / 377214ØØØØØØØØØØØØØØØB /
```

```
C     DATA RMACH(4) / 377224ØØØØØØØØØØØØØØØB /
C     DATA RMACH(5) / 377774642Ø23241175720B /
C
C     MACHINE CONSTANTS FOR THE DATA GENERAL ECLIPSE S/2ØØ                MCHR246Ø
C                                                                         MCHR248Ø
C     NOTE - IT MAY BE APPROPRIATE TO INCLUDE THE FOLLOWING CARD -        MCHR25ØØ
C     STATIC RMACH(5)                                                     MCHR252Ø
C                                                                         MCHR254Ø
C     DATA SMALL/2ØK,Ø/,LARGE/77777K,177777K/                             MCHR256Ø
C     DATA RIGHT/3542ØK,Ø/,DIVER/36Ø2ØK,Ø/                                MCHR258Ø
C     DATA LOG1Ø/4Ø423K,42Ø23K/                                           MCHR26ØØ
C                                                                         MCHR262Ø
C     MACHINE CONSTANTS FOR THE HARRIS 22Ø                                MCHR264Ø
C                                                                         MCHR266Ø
C     DATA SMALL(1),SMALL(2) / '2ØØØØØØØ, 'ØØØØØ2Ø1 /                    MCHR268Ø
C     DATA LARGE(1),LARGE(2) / '37777777, 'ØØØØØ177 /                     MCHR27ØØ
C     DATA RIGHT(1),RIGHT(2) / '2ØØØØØØØ, 'ØØØØØ352 /                     MCHR272Ø
C     DATA DIVER(1),DIVER(2) / '2ØØØØØØØ, 'ØØØØØ353 /                     MCHR274Ø
C     DATA LOG1Ø(1),LOG1Ø(2) / '2321Ø115, 'ØØØØØ377 /                     MCHR276Ø
C                                                                         MCHR278Ø
C     MACHINE CONSTANTS FOR THE HONEYWELL 6ØØ/6ØØØ SERIES.                MCHRØ72Ø
C                                                                         MCHRØ74Ø
C     DATA RMACH(1) / O4Ø24ØØØØØØØØ /                                     MCHRØ76Ø
C     DATA RMACH(2) / O376777777777 /                                     MCHRØ78Ø
C     DATA RMACH(3) / O7144ØØØØØØØØ /                                     MCHRØ8ØØ
C     DATA RMACH(4) / O7164ØØØØØØØØ /                                     MCHRØ82Ø
C     DATA RMACH(5) / O776464Ø2324 /                                      MCHRØ84Ø
C                                                                         MCHRØ86Ø
C     MACHINE CONSTANTS FOR THE IBM 36Ø/37Ø SERIES,                       MCHRØ88Ø
C     THE XEROX SIGMA 5/7/9 AND THE SEL SYSTEMS 85/86.                    MCHRØ9ØØ
C                                                                         MCHRØ92Ø
C     DATA RMACH(1) / Z4Ø1ØØØØØ /                                         MCHRØ94Ø
C     DATA RMACH(2) / Z7FFFFFFF /                                         MCHRØ96Ø
C     DATA RMACH(3) / Z3B1ØØØØØ /                                         MCHRØ98Ø
C     DATA RMACH(4) / Z3C1ØØØØØ /                                         MCHR1ØØØ
C     DATA RMACH(5) / Z41134413 /                                         MCHR1Ø2Ø
C                                                                         MCHR1Ø4Ø
C     MACHINE CONSTANTS FOR THE PDP-1Ø (KA OR KI PROCESSOR).              MCHR122Ø
C                                                                         MCHR124Ø
C     DATA RMACH(1) / "ØØØ4ØØØØØØØØ /                                     MCHR126Ø
C     DATA RMACH(2) / "377777777777 /                                     MCHR128Ø
C     DATA RMACH(3) / "1464ØØØØØØØØ /                                     MCHR13ØØ
C     DATA RMACH(4) / "1474ØØØØØØØØ /                                     MCHR132Ø
C     DATA RMACH(5) / "177464Ø2324 /                                      MCHR134Ø
C                                                                         MCHR136Ø
C     MACHINE CONSTANTS FOR PDP-11 FORTRAN'S SUPPORTING                   MCHR138Ø
C     32-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).                   MCHR14ØØ
C                                                                         MCHR142Ø
C     DATA SMALL(1) /    8388608 /                                        MCHR144Ø
C     DATA LARGE(1) / 2147483647 /                                        MCHR146Ø
C     DATA RIGHT(1) /  88Ø8Ø384Ø /                                        MCHR148Ø
C     DATA DIVER(1) /  889192448 /                                        MCHR15ØØ
C     DATA LOG1Ø(1) / 1Ø67Ø65499 /                                        MCHR152Ø
C                                                                         MCHR154Ø
C     DATA RMACH(1) / OØØØ4ØØØØØØØ /                                      MCHR156Ø
C     DATA RMACH(2) / O17777777777 /                                      MCHR158Ø
C     DATA RMACH(3) / OØ644ØØØØØØØ /                                      MCHR16ØØ
C     DATA RMACH(4) / OØ65ØØØØØØØØ /                                      MCHR162Ø
C     DATA RMACH(5) / OØ7746Ø2233 /                                       MCHR164Ø
C                                                                         MCHR166Ø
C     MACHINE CONSTANTS FOR PDP-11 FORTRAN'S SUPPORTING                   MCHR168Ø
C     16-BIT INTEGERS  (EXPRESSED IN INTEGER AND OCTAL).                  MCHR17ØØ
C                                                                         MCHR172Ø
C     DATA SMALL(1),SMALL(2) /   128,     Ø /                             MCHR174Ø
C     DATA LARGE(1),LARGE(2) / 32767,    -1 /                             MCHR176Ø
C     DATA RIGHT(1),RIGHT(2) / 1344Ø,     Ø /                             MCHR178Ø
C     DATA DIVER(1),DIVER(2) / 13568,     Ø /                             MCHR18ØØ
C     DATA LOG1Ø(1),LOG1Ø(2) / 16282,  8347 /                             MCHR182Ø
C                                                                         MCHR184Ø
C     DATA SMALL(1),SMALL(2) / OØØØ2ØØ, OØØØØØØ /                         MCHR186Ø
C     DATA LARGE(1),LARGE(2) / OØ77777, O177777 /                         MCHR188Ø
C     DATA RIGHT(1),RIGHT(2) / OØ322ØØ, OØØØØØØ /                         MCHR19ØØ
C     DATA DIVER(1),DIVER(2) / OØ324ØØ, OØØØØØØ /                         MCHR192Ø
C     DATA LOG1Ø(1),LOG1Ø(2) / OØ37632, OØ2Ø233 /                         MCHR194Ø
C                                                                         MCHR196Ø
```

```
C     MACHINE CONSTANTS FOR THE UNIVAC 11ØØ SERIES.                    MCHR23ØØ
C                                                                      MCHR232Ø
C     DATA RMACH(1) / OØØØ4ØØØØØØØ /                                   MCHR234Ø
C     DATA RMACH(2) / O377777777777 /                                  MCHR236Ø
C     DATA RMACH(3) / O1464ØØØØØØØ /                                   MCHR238Ø
C     DATA RMACH(4) / O1474ØØØØØØØ /                                   MCHR24ØØ
C     DATA RMACH(5) / O177464Ø2324 /                                   MCHR242Ø
C                                                                      MCHR244Ø
      IF (I .LT. 1  .OR.  I .GT. 5)                                    MCHR28ØØ
     1   CALL SETERR(24HR1MACH - I OUT OF BOUNDS,24,1,2)               MCHR282Ø
C                                                                      MCHR284Ø
      R1MACH = RMACH(I)                                                MCHR286Ø
      RETURN                                                           MCHR288Ø
C                                                                      MCHR29ØØ
      END                                                              MCHR292Ø
```

The function for DOUBLE-PRECISION floating-point numbers is

```
      DOUBLE PRECISION FUNCTION D1MACH(I)                              MCHDØØØØ
C                                                                      MCHDØØ2Ø
C  DOUBLE-PRECISION MACHINE CONSTANTS                                  MCHDØØ4Ø
C                                                                      MCHDØØ6Ø
C  D1MACH( 1) = B**(EMIN-1), THE SMALLEST POSITIVE MAGNITUDE.          MCHDØØ8Ø
C                                                                      MCHDØ1ØØ
C  D1MACH( 2) = B**EMAX*(1 - B**(-T)), THE LARGEST MAGNITUDE.         MCHDØ12Ø
C                                                                      MCHDØ14Ø
C  D1MACH( 3) = B**(-T), THE SMALLEST RELATIVE SPACING.                MCHDØ16Ø
C                                                                      MCHDØ18Ø
C  D1MACH( 4) = B**(1-T), THE LARGEST RELATIVE SPACING.                MCHDØ2ØØ
C                                                                      MCHDØ22Ø
C  D1MACH( 5) = LOG1Ø(B)                                               MCHDØ24Ø
C                                                                      MCHDØ26Ø
C  TO ALTER THIS FUNCTION FOR A PARTICULAR ENVIRONMENT,                MCHDØ28Ø
C  THE DESIRED SET OF DATA STATEMENTS SHOULD BE ACTIVATED BY           MCHDØ3ØØ
C  REMOVING THE C FROM COLUMN 1.                                       MCHDØ32Ø
C                                                                      MCHDØ34Ø
C  WHERE POSSIBLE, OCTAL OR HEXADECIMAL CONSTANTS HAVE BEEN USED       MCHDØ36Ø
C  TO SPECIFY THE CONSTANTS EXACTLY WHICH HAS IN SOME CASES            MCHDØ38Ø
C  REQUIRED THE USE OF EQUIVALENT INTEGER ARRAYS.                      MCHDØ4ØØ
C                                                                      MCHDØ42Ø
      INTEGER SMALL(4)                                                 MCHDØ44Ø
      INTEGER LARGE(4)                                                 MCHDØ46Ø
      INTEGER RIGHT(4)                                                 MCHDØ48Ø
      INTEGER DIVER(4)                                                 MCHDØ5ØØ
      INTEGER LOG1Ø(4)                                                 MCHDØ52Ø
C                                                                      MCHDØ54Ø
      DOUBLE PRECISION DMACH(5)                                        MCHDØ56Ø
C                                                                      MCHDØ58Ø
      EQUIVALENCE (DMACH(1),SMALL(1))                                  MCHDØ6ØØ
      EQUIVALENCE (DMACH(2),LARGE(1))                                  MCHDØ62Ø
      EQUIVALENCE (DMACH(3),RIGHT(1))                                  MCHDØ64Ø
      EQUIVALENCE (DMACH(4),DIVER(1))                                  MCHDØ66Ø
      EQUIVALENCE (DMACH(5),LOG1Ø(1))                                  MCHDØ68Ø
C                                                                      MCHDØ7ØØ
C     MACHINE CONSTANTS FOR THE BURROUGHS 17ØØ SYSTEM.                 MCHD336Ø
C                                                                      MCHD338Ø
C     DATA SMALL(1) / ZCØØ8ØØØØØ /                                     MCHD34ØØ
C     DATA SMALL(2) / ZØØØØØØØØØ /                                     MCHD342Ø
C                                                                      MCHD344Ø
C     DATA LARGE(1) / ZDFFFFFFFF /                                     MCHD346Ø
C     DATA LARGE(2) / ZFFFFFFFFF /                                     MCHD348Ø
C                                                                      MCHD35ØØ
C     DATA RIGHT(1) / ZCC58ØØØØØ /                                     MCHD352Ø
C     DATA RIGHT(2) / ZØØØØØØØØØ /                                     MCHD354Ø
C                                                                      MCHD356Ø
C     DATA DIVER(1) / ZCC68ØØØØØ /                                     MCHD358Ø
C     DATA DIVER(2) / ZØØØØØØØØØ /                                     MCHD36ØØ
C                                                                      MCHD362Ø
C     DATA LOG1Ø(1) / ZDØØE73ØE7 /                                     MCHD364Ø
C     DATA LOG1Ø(2) / ZC778ØØDCØ /                                     MCHD366Ø
C                                                                      MCHD368Ø
C     MACHINE CONSTANTS FOR THE BURROUGHS 57ØØ SYSTEM.                 MCHD3Ø2Ø
C                                                                      MCHD3Ø4Ø
C     DATA SMALL(1) / O1771ØØØØØØØØØØØ /                               MCHD3Ø6Ø
C     DATA SMALL(2) / OØØØØØØØØØØØØØØØØ /                             MCHD3Ø8Ø
C                                                                      MCHD31ØØ
```

```
C     DATA LARGE(1) / O0777777777777777 /                                MCHD312Ø
C     DATA LARGE(2) / O000777777777777 /                                 MCHD314Ø
C                                                                        MCHD316Ø
C     DATA RIGHT(1) / O14610000000000000 /                               MCHD318Ø
C     DATA RIGHT(2) / O0000000000000000 /                                MCHD32ØØ
C                                                                        MCHD322Ø
C     DATA DIVER(1) / O14510000000000000 /                               MCHD324Ø
C     DATA DIVER(2) / O0000000000000000 /                                MCHD326Ø
C                                                                        MCHD328Ø
C     DATA LOG1Ø(1) / O1157163Ø34761674 /                                MCHD33ØØ
C     DATA LOG1Ø(2) / O0006677466732724 /                                MCHD332Ø
C                                                                        MCHD334Ø
C     MACHINE CONSTANTS FOR THE BURROUGHS 67ØØ/77ØØ SYSTEMS.             MCHD268Ø
C                                                                        MCHD27ØØ
C     DATA SMALL(1) / O1771000000000000 /                                MCHD272Ø
C     DATA SMALL(2) / O0770000000000000 /                                MCHD274Ø
C                                                                        MCHD276Ø
C     DATA LARGE(1) / O0777777777777777 /                                MCHD278Ø
C     DATA LARGE(2) / O7777777777777777 /                                MCHD28ØØ
C                                                                        MCHD282Ø
C     DATA RIGHT(1) / O14610000000000000 /                               MCHD284Ø
C     DATA RIGHT(2) / O0000000000000000 /                                MCHD286Ø
C                                                                        MCHD288Ø
C     DATA DIVER(1) / O14510000000000000 /                               MCHD29ØØ
C     DATA DIVER(2) / O0000000000000000 /                                MCHD292Ø
C                                                                        MCHD294Ø
C     DATA LOG1Ø(1) / O1157163Ø34761674 /                                MCHD296Ø
C     DATA LOG1Ø(2) / O0006677466732724 /                                MCHD298Ø
C                                                                        MCHD3ØØØ
C     MACHINE CONSTANTS FOR THE CDC 6ØØØ/7ØØØ SERIES.                    MCHD1Ø6Ø
C                                                                        MCHD1Ø8Ø
C     DATA SMALL(1) / ØØ6Ø4ØØØØØØØØØØØØØØØB /                           MCHD11ØØ
C     DATA SMALL(2) / ØØØØØØØØØØØØØØØØØØØB /                            MCHD112Ø
C                                                                        MCHD114Ø
C     DATA LARGE(1) / 377677777777777777B /                              MCHD116Ø
C     DATA LARGE(2) / 371677777777777777B /                              MCHD118Ø
C                                                                        MCHD12ØØ
C     DATA RIGHT(1) / 1564ØØØØØØØØØØØØØØØB /                            MCHD122Ø
C     DATA RIGHT(2) / 15ØØØØØØØØØØØØØØØØØB /                            MCHD124Ø
C                                                                        MCHD126Ø
C     DATA DIVER(1) / 1561ØØØØØØØØØØØØØØØB /                            MCHD128Ø
C     DATA DIVER(2) / 15Ø1ØØØØØØØØØØØØØØØB /                            MCHD13ØØ
C                                                                        MCHD132Ø
C     DATA LOG1Ø(1) / 1716464ØØ23241175717B /                            MCHD134Ø
C     DATA LOG1Ø(2) / 16367571421742254654B /                            MCHD136Ø
C                                                                        MCHD138Ø
C     MACHINE CONSTANTS FOR THE CRAY 1
C
C     DATA SMALL(1) / 2ØØØØ4ØØØØØØØØØØØØØØØB /
C     DATA SMALL(2) / ØØØØØØØØØØØØØØØØØØØB /
C
C     DATA LARGE(1) / 577777777777777777B /
C     DATA LARGE(2) / ØØØØØ7777777777777777B /
C     DATA RIGHT(1) / 377214ØØØØØØØØØØØØØØØB /
C     DATA RIGHT(2) / ØØØØØØØØØØØØØØØØØØØØB /
C
C     DATA DIVER(1) / 377224ØØØØØØØØØØØØØØØB /
C     DATA DIVER(2) / ØØØØØØØØØØØØØØØØØØØØB /
C
C     DATA LOG1Ø(1) / 37777464ØØ23241175717B /
C     DATA LOG1Ø(2) / ØØØØØ75714217422546548 /
C
C     MACHINE CONSTANTS FOR THE DATA GENERAL ECLIPSE S/2ØØ               MCHD388Ø
C                                                                        MCHD39ØØ
C     NOTE - IT MAY BE APPROPRIATE TO INCLUDE THE FOLLOWING CARD -       MCHD392Ø
C     STATIC DMACH(5)                                                    MCHD394Ø
C                                                                        MCHD396Ø
C     DATA SMALL/2ØK,3*Ø/,LARGE/77777K,3*177777K/                        MCHD398Ø
C     DATA RIGHT/3142ØK,3*Ø/,DIVER/32Ø2ØK,3*Ø/                           MCHD4ØØØ
C     DATA LOG1Ø/4Ø423K,42Ø23K,5Ø237K,74776K/                            MCHD4Ø2Ø
C                                                                        MCHD4Ø4Ø
C     MACHINE CONSTANTS FOR THE HARRIS 22Ø                               MCHD4Ø6Ø
C                                                                        MCHD4Ø8Ø
C     DATA SMALL(1),SMALL(2) / '2ØØØØØØØ, 'ØØØØØ2Ø1 /                   MCHD41ØØ
C     DATA LARGE(1),LARGE(2) / '37777777, '37777577 /                    MCHD412Ø
```

```
C     DATA RIGHT(1),RIGHT(2) / '2ØØØØØØØ, 'ØØØØØ333 /                MCHD414Ø
C     DATA DIVER(1),DIVER(2) / '2ØØØØØØØ, 'ØØØØØ334 /                MCHD416Ø
C     DATA LOG1Ø(1),LOG1Ø(2) / '2321Ø115, '1Ø237777 /                MCHD418Ø
C
C     MACHINE CONSTANTS FOR THE HONEYWELL 6ØØ/6ØØØ SERIES.           MCHDØ72Ø
C                                                                    MCHDØ74Ø
C     DATA SMALL(1),SMALL(2) / O4Ø24ØØØØØØØ, OØØØØØØØØØØØ /         MCHDØ76Ø
C     DATA LARGE(1),LARGE(2) / O376777777777, O777777777777 /        MCHDØ78Ø
C     DATA RIGHT(1),RIGHT(2) / O6Ø44ØØØØØØØ, OØØØØØØØØØØØ /         MCHDØ8ØØ
C     DATA DIVER(1),DIVER(2) / O6Ø64ØØØØØØØ, OØØØØØØØØØØØ /         MCHDØ82Ø
C     DATA LOG1Ø(1),LOG1Ø(2) / O776464Ø2324, O117571775714 /         MCHDØ84Ø
C                                                                    MCHDØ86Ø
C     MACHINE CONSTANTS FOR THE IBM 36Ø/37Ø SERIES,                  MCHDØ88Ø
C     THE XEROX SIGMA 5/7/9 AND THE SEL SYSTEMS 85/86.               MCHDØ9ØØ
C                                                                    MCHDØ92Ø
C     DATA SMALL(1),SMALL(2) / ZØØ1ØØØØØ, ZØØØØØØØØ /               MCHDØ94Ø
C     DATA LARGE(1),LARGE(2) / Z7FFFFFFF, ZFFFFFFFF /                MCHDØ96Ø
C     DATA RIGHT(1),RIGHT(2) / Z331ØØØØØ, ZØØØØØØØØ /               MCHDØ98Ø
C     DATA DIVER(1),DIVER(2) / Z341ØØØØØ, ZØØØØØØØØ /               MCHD1ØØØ
C     DATA LOG1Ø(1),LOG1Ø(2) / Z41134413, Z5Ø9F79FF /                MCHD1Ø2Ø
C                                                                    MCHD1Ø4Ø
C     MACHINE CONSTANTS FOR THE PDP-1Ø (KA PROCESSOR).               MCHD14ØØ
C                                                                    MCHD142Ø
C     DATA SMALL(1),SMALL(2) / "Ø334ØØØØØØØ, "ØØØØØØØØØØØ /         MCHD144Ø
C     DATA LARGE(1),LARGE(2) / "377777777777, "344777777777 /        MCHD146Ø
C     DATA RIGHT(1),RIGHT(2) / "1134ØØØØØØØ, "ØØØØØØØØØØØ /         MCHD148Ø
C     DATA DIVER(1),DIVER(2) / "1144ØØØØØØØ, "ØØØØØØØØØØØ /         MCHD15ØØ
C     DATA LOG1Ø(1),LOG1Ø(2) / "177464Ø2324, "144117571776 /         MCHD152Ø
C                                                                    MCHD154Ø
C     MACHINE CONSTANTS FOR THE PDP-1Ø (KI PROCESSOR).               MCHD156Ø
C                                                                    MCHD158Ø
C     DATA SMALL(1),SMALL(2) / "ØØØ4ØØØØØØØ, "ØØØØØØØØØØØ /         MCHD16ØØ
C     DATA LARGE(1),LARGE(2) / "377777777777, "377777777777 /        MCHD162Ø
C     DATA RIGHT(1),RIGHT(2) / "1Ø34ØØØØØØØ, "ØØØØØØØØØØØ /         MCHD164Ø
C     DATA DIVER(1),DIVER(2) / "1Ø44ØØØØØØØ, "ØØØØØØØØØØØ /         MCHD166Ø
C     DATA LOG1Ø(1),LOG1Ø(2) / "177464Ø2324, "476747767461 /         MCHD168Ø
C                                                                    MCHD17ØØ
C     MACHINE CONSTANTS FOR PDP-11 FORTRAN'S SUPPORTING              MCHD172Ø
C     32-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).              MCHD174Ø
C                                                                    MCHD176Ø
C     DATA SMALL(1),SMALL(2) /    8388608,          Ø /              MCHD178Ø
C     DATA LARGE(1),LARGE(2) / 2147483647,         -1 /              MCHD18ØØ
C     DATA RIGHT(1),RIGHT(2) /  612368384,          Ø /              MCHD182Ø
C     DATA DIVER(1),DIVER(2) /  62Ø756992,          Ø /              MCHD184Ø
C     DATA LOG1Ø(1),LOG1Ø(2) / 1Ø67Ø65498, -2Ø63872ØØ8 /             MCHD186Ø
C                                                                    MCHD188Ø
C     DATA SMALL(1),SMALL(2) / OØØØ4ØØØØØØ, OØØØØØØØØØØ /          MCHD19ØØ
C     DATA LARGE(1),LARGE(2) / O17777777777, O37777777777 /         MCHD192Ø
C     DATA RIGHT(1),RIGHT(2) / OØ444ØØØØØØ, OØØØØØØØØØØ /          MCHD194Ø
C     DATA DIVER(1),DIVER(2) / OØ45ØØØØØØØ, OØØØØØØØØØØ /          MCHD196Ø
C     DATA LOG1Ø(1),LOG1Ø(2) / OØ7746Ø2Ø232, O2Ø476747770 /          MCHD198Ø
C                                                                    MCHD2ØØØ
C     MACHINE CONSTANTS FOR PDP-11 FORTRAN'S SUPPORTING              MCHD2Ø2Ø
C     16-BIT INTEGERS (EXPRESSED IN INTEGER AND OCTAL).              MCHD2Ø4Ø
C                                                                    MCHD2Ø6Ø
C     DATA SMALL(1),SMALL(2) /    128,      Ø /                      MCHD2Ø8Ø
C     DATA SMALL(3),SMALL(4) /      Ø,      Ø /                      MCHD21ØØ
C                                                                    MCHD212Ø
C     DATA LARGE(1),LARGE(2) /  32767,     -1 /                      MCHD214Ø
C     DATA LARGE(3),LARGE(4) /     -1,     -1 /                      MCHD216Ø
C                                                                    MCHD218Ø
C     DATA RIGHT(1),RIGHT(2) /   9344,      Ø /                      MCHD22ØØ
C     DATA RIGHT(3),RIGHT(4) /      Ø,      Ø /                      MCHD222Ø
C                                                                    MCHD224Ø
C     DATA DIVER(1),DIVER(2) /   9472,      Ø /                      MCHD226Ø
C     DATA DIVER(3),DIVER(4) /      Ø,      Ø /                      MCHD228Ø
C                                                                    MCHD23ØØ
C     DATA LOG1Ø(1),LOG1Ø(2) /  16282,   8346 /                      MCHD232Ø
C     DATA LOG1Ø(3),LOG1Ø(4) / -31493, -12296 /                      MCHD234Ø
C                                                                    MCHD236Ø
C     DATA SMALL(1),SMALL(2) / OØØØ2ØØ, OØØØØØØ /                   MCHD238Ø
C     DATA SMALL(3),SMALL(4) / OØØØØØØ, OØØØØØØ /                   MCHD24ØØ
C                                                                    MCHD242Ø
C     DATA LARGE(1),LARGE(2) / OØ77777, O177777 /                    MCHD244Ø
C     DATA LARGE(3),LARGE(4) / O177777, O177777 /                    MCHD246Ø
C                                                                    MCHD248Ø
```

```
C     DATA RIGHT(1),RIGHT(2) / 0Ø222ØØ, 0ØØØØØØ /                    MCHD25ØØ
C     DATA RIGHT(3),RIGHT(4) / 0ØØØØØØ, 0ØØØØØØ /                    MCHD252Ø
C                                                                     MCHD254Ø
C     DATA DIVER(1),DIVER(2) / 0Ø224ØØ, 0ØØØØØØ /                    MCHD256Ø
C     DATA DIVER(3),DIVER(4) / 0ØØØØØØ, 0ØØØØØØ /                    MCHD258Ø
C                                                                     MCHD26ØØ
C     DATA LOG1Ø(1),LOG1Ø(2) / 0Ø37632, 0Ø2Ø232 /                    MCHD262Ø
C     DATA LOG1Ø(3),LOG1Ø(4) / 01Ø2373, 014777Ø /                    MCHD264Ø
C                                                                     MCHD266Ø
C     MACHINE CONSTANTS FOR THE UNIVAC 11ØØ SERIES.                  MCHD37ØØ
C                                                                     MCHD372Ø
C     DATA SMALL(1),SMALL(2) / 0ØØØØ4ØØØØØØØ, 0ØØØØØØØØØØØØ /      MCHD374Ø
C     DATA LARGE(1),LARGE(2) / 0377777777777, 0777777777777 /       MCHD376Ø
C     DATA RIGHT(1),RIGHT(2) / 017Ø54ØØØØØØØ, 0ØØØØØØØØØØØØ /      MCHD378Ø
C     DATA DIVER(1),DIVER(2) / 017Ø64ØØØØØØØ, 0ØØØØØØØØØØØØ /      MCHD38ØØ
C     DATA LOG1Ø(1),LOG1Ø(2) / 017774642Ø232, 0411757177572 /       MCHD382Ø
C                                                                     MCHD386Ø
C                                                                     MCHD42ØØ
      IF (I .LT. 1  .OR.  I .GT. 5)                                  MCHD422Ø
     1   CALL SETERR(24HD1MACH - I OUT OF BOUNDS,24,1,2)             MCHD424Ø
C                                                                     MCHD426Ø
      D1MACH = DMACH(I)                                              MCHD428Ø
      RETURN                                                         MCHD43ØØ
C                                                                     MCHD432Ø
      END                                                            MCHD434Ø
```

Automatic Error Handling

The second package provides a basic mechanism for dealing with the occurrence of errors.

In the PORT library [1], for which the package was developed, calls to the general subroutines in the library do not include flags for error indication in their calling sequences. Instead, when a called subroutine detects an error, it calls the principal error-handling routine, SETERR.

The package allows for two types of error, "fatal," and "recoverable," and a parameter in the call to SETERR must be set to specify the type. Fatal errors cause an error message to be printed, the run terminated, and a call made to a dump routine. (A dummy dump routine, FDUMP, is provided here.) For recoverable errors, unless the user has specifically requested to enter the recovery mode, similar events occur, an error message is printed, and the run terminated. Thus the process is failsafe for unwary users.

When the recovery mode is in effect, any call to SETERR given within a subprogram which has detected a recoverable error has the effect only of storing the fact that an error has occurred; the run is not terminated. The user, upon return from the subprogram, is responsible for testing for the occurrence of an error. If an error has occurred, the user must turn off the error state, because additional errors might arise and the occurrence of a recoverable error while in the error state constitutes an unrecoverable error, terminating the run.

Finally, since a called subprogram, say SUBA, may, in turn, call a lower-level subprogram containing recoverable errors, SUBA must check for the occurrence of errors in the lower-level routine and reinterpret them in the context of SUBA, which the user knows about. This means that SUBA must enter the recovery mode (saving the mode previously in effect), make the call to the lower-level subprogram, then, upon return from the lower-level routine, check for errors, and, before returning to the user, restore the previous recovery mode.

An error which has caused an invocation of SETERR has an associated number, message, and type (fatal or recoverable), and the effect of the error depends on whether the recovery mode is in effect or not. The various capabilities offered in the subprograms of the package are summarized as follows.

To signal that an error has occurred:

CALL SETERR(MESSG, NMESSG, NERR, IOPT)

where MESSG and NMESSG are, respectively, a Hollerith message and the number of characters in the message, and NERR is the error number. IOPT is used to specify the type of error: IOPT = 1 for a recoverable error, and IOPT = 2 for a fatal error.

To save the recovery (or nonrecovery) mode currently in effect, and enter a new one:

```
CALL ENTSRC(IROLD, IRNEW)
```

which saves the current mode in IROLD and sets the new one to IRNEW.

To avoid having multiple errors outstanding, it is a fatal error to call SETERR or ENTSRC if the error state is on, meaning that an error has occurred but has not been recovered from.

To restore the recovery (or nonrecovery) mode which was previously saved in IROLD:

```
CALL RETSRC(IROLD)
```

where RETSRC not only restores the previous mode, but also acts as a "safety" exit gate: Since multiple errors are illegal, RETSRC checks out the situation and allows return to the calling program only if (1) an error is not outstanding, or (2) the restored mode is recovery, so that the calling program is responsible for error checking.

To test if an error has occurred, and if its number was, say, 4, a statement such as the following is used:

```
IF (NERROR(NERR) .EQ. 4) GO TO 50
```

The value of the function NERROR and the value of the argument NERR are both set to the current value of the error number by NERROR. (The double assignment may be useful and comes free since Fortran prohibits functions with no arguments.) If the error number is nonzero, it means that an error has occurred and that corrective action must be taken.

To turn off the error state:

```
CALL ERROFF
```

In summary the user subprograms are:

SETERR	turns on the error state and saves a message and an error number;
ENTSRC	at entry, sets recovery (or nonrecovery) mode, provided no error state exists;
RETSRC	before returning, checks error situation and, if no errors exist, restores prior recovery (or nonrecovery) mode;
NERROR	returns the error number;
ERROFF	turns off the error state;
EPRINT	prints the error message.

These, in turn, call on the lower-level subprograms:

E9RINT	stores or prints error message, depending on switch setting;
S88FMT	sets up FORMAT array for printing;
I8SAVE	returns error number or recovery (or nonrecovery) mode, depending on one switch, and resets or does not reset the corresponding value depending on another;
FDUMP	a dummy routine to be replaced, if possible, by a locally written symbolic dump routine.

ALGORITHM

```
      SUBROUTINE SETERR(MESSG,NMESSG,NERR,IOPT)                          ERRSØØØØ
C                                                                        ERRSØØ2Ø
C  SETERR SETS LERROR = NERR, OPTIONALLY PRINTS THE MESSAGE AND DUMPS    ERRSØØ4Ø
C  ACCORDING TO THE FOLLOWING RULES...                                   ERRSØØ6Ø
C                                                                        ERRSØØ8Ø
C    IF IOPT = 1 AND RECOVERING      - JUST REMEMBER THE ERROR.          ERRSØ1ØØ
C    IF IOPT = 1 AND NOT RECOVERING  - PRINT AND STOP.                   ERRSØ12Ø
C    IF IOPT = 2                     - PRINT, DUMP AND STOP.             ERRSØ14Ø
C                                                                        ERRSØ16Ø
C  INPUT                                                                 ERRSØ18Ø
C                                                                        ERRSØ2ØØ
C    MESSG  - THE ERROR MESSAGE.                                         ERRSØ22Ø
C    NMESSG - THE LENGTH OF THE MESSAGE, IN CHARACTERS.                  ERRSØ24Ø
C    NERR   - THE ERROR NUMBER. MUST HAVE NERR NON-ZERO.                 ERRSØ26Ø
C    IOPT   - THE OPTION. MUST HAVE IOPT=1 OR 2.                         ERRSØ28Ø
C                                                                        ERRSØ3ØØ
C  ERROR STATES -                                                        ERRSØ32Ø
C                                                                        ERRSØ34Ø
C    1 - MESSAGE LENGTH NOT POSITIVE.                                    ERRSØ36Ø
C    2 - CANNOT HAVE NERR=Ø.                                             ERRSØ38Ø
C    3 - AN UNRECOVERED ERROR FOLLOWED BY ANOTHER ERROR.                 ERRSØ4ØØ
C    4 - BAD VALUE FOR IOPT.                                             ERRSØ42Ø
C                                                                        ERRSØ44Ø
C  ONLY THE FIRST 72 CHARACTERS OF THE MESSAGE ARE PRINTED.              ERRSØ46Ø
C                                                                        ERRSØ48Ø
C  THE ERROR HANDLER CALLS A SUBROUTINE NAMED FDUMP TO PRODUCE A         ERRSØ5ØØ
C  SYMBOLIC DUMP. TO COMPLETE THE PACKAGE, A DUMMY VERSION OF FDUMP      ERRSØ52Ø
C  IS SUPPLIED, BUT IT SHOULD BE REPLACED BY A LOCALLY WRITTEN VERSION   ERRSØ54Ø
C  WHICH AT LEAST GIVES A TRACE-BACK.                                    ERRSØ56Ø
C                                                                        ERRSØ58Ø
      INTEGER MESSG(1)                                                   ERRSØ6ØØ
C                                                                        ERRSØ62Ø
C  THE UNIT FOR ERROR MESSAGES.                                          ERRSØ64Ø
C                                                                        ERRSØ66Ø
      IWUNIT=I1MACH(4)                                                   ERRSØ68Ø
C                                                                        ERRSØ7ØØ
      IF (NMESSG.GE.1) GO TO 1Ø                                          ERRSØ72Ø
C                                                                        ERRSØ74Ø
C  A MESSAGE OF NON-POSITIVE LENGTH IS FATAL.                            ERRSØ76Ø
C                                                                        ERRSØ78Ø
      WRITE(IWUNIT,9ØØØ)                                                 ERRSØ8ØØ
 9ØØØ FORMAT(52H1ERROR    1 IN SETERR - MESSAGE LENGTH NOT POSITIVE.)    ERRSØ82Ø
      GO TO 6Ø                                                           ERRSØ84Ø
C                                                                        ERRSØ86Ø
C  NW IS THE NUMBER OF WORDS THE MESSAGE OCCUPIES.                       ERRSØ88Ø
C                                                                        ERRSØ9ØØ
 1Ø   NW=(MINØ(NMESSG,72)-1)/I1MACH(6)+1                                 ERRSØ92Ø
C                                                                        ERRSØ94Ø
      IF (NERR.NE.Ø) GO TO 2Ø                                            ERRSØ96Ø
C                                                                        ERRSØ98Ø
C  CANNOT TURN THE ERROR STATE OFF USING SETERR.                         ERRS1ØØØ
C                                                                        ERRS1Ø2Ø
      WRITE(IWUNIT,9ØØ1)                                                 ERRS1Ø4Ø
 9ØØ1 FORMAT(42H1ERROR    2 IN SETERR - CANNOT HAVE NERR=Ø//            ERRS1Ø6Ø
     1        34H THE CURRENT ERROR MESSAGE FOLLOWS///)                  ERRS1Ø8Ø
      CALL E9RINT(MESSG,NW,NERR,.TRUE.)                                  ERRS11ØØ
      ITEMP=I8SAVE(1,1,.TRUE.)                                           ERRS112Ø
      GO TO 5Ø                                                           ERRS114Ø
C                                                                        ERRS116Ø
C  SET LERROR AND TEST FOR A PREVIOUS UNRECOVERED ERROR.                 ERRS118Ø
C                                                                        ERRS12ØØ
 2Ø   IF (I8SAVE(1,NERR,.TRUE.).EQ.Ø) GO TO 3Ø                           ERRS122Ø
C                                                                        ERRS124Ø
      WRITE(IWUNIT,9ØØ2)                                                 ERRS126Ø
 9ØØ2 FORMAT(23H1ERROR    3 IN SETERR -,                                ERRS128Ø
     1        48H AN UNRECOVERED ERROR FOLLOWED BY ANOTHER ERROR.//     ERRS13ØØ
     2        48H THE PREVIOUS AND CURRENT ERROR MESSAGES FOLLOW.///)   ERRS132Ø
      CALL EPRINT                                                        ERRS134Ø
      CALL E9RINT(MESSG,NW,NERR,.TRUE.)                                  ERRS136Ø
      GO TO 5Ø                                                           ERRS138Ø
C                                                                        ERRS14ØØ
C  SAVE THIS MESSAGE IN CASE IT IS NOT RECOVERED FROM PROPERLY.          ERRS142Ø
```

```
C                                                                       ERRS144Ø
 3Ø   CALL E9RINT(MESSG,NW,NERR,.TRUE.)                                 ERRS146Ø
C                                                                       ERRS148Ø
      IF (IOPT.EQ.1 .OR. IOPT.EQ.2) GO TO 4Ø                            ERRS15ØØ
C                                                                       ERRS152Ø
C  MUST HAVE IOPT = 1 OR 2.                                             ERRS154Ø
C                                                                       ERRS156Ø
        WRITE(IWUNIT,9ØØ3)                                              ERRS158Ø
 9ØØ3   FORMAT(42H1ERROR    4 IN SETERR - BAD VALUE FOR IOPT//           ERRS16ØØ
     1         34H THE CURRENT ERROR MESSAGE FOLLOWS///)                ERRS162Ø
        GO TO 5Ø                                                        ERRS164Ø
C                                                                       ERRS166Ø
C  TEST FOR RECOVERY.                                                   ERRS168Ø
C                                                                       ERRS17ØØ
 4Ø   IF (IOPT.EQ.2) GO TO 5Ø                                           ERRS172Ø
C                                                                       ERRS174Ø
      IF (I8SAVE(2,Ø,.FALSE.).EQ.1) RETURN                              ERRS176Ø
C                                                                       ERRS178Ø
      CALL EPRINT                                                       ERRS18ØØ
      STOP                                                              ERRS182Ø
C                                                                       ERRS184Ø
 5Ø   CALL EPRINT                                                       ERRS186Ø
 6Ø   CALL FDUMP                                                        ERRS188Ø
      STOP                                                              ERRS19ØØ
C                                                                       ERRS192Ø
      END                                                               ERRS194Ø


      SUBROUTINE ENTSRC(IROLD,IRNEW)                                    RECAØØØØ
C                                                                       RECAØØ2Ø
C  THIS ROUTINE RETURNS IROLD = LRECOV AND SETS LRECOV = IRNEW.          RECAØØ4Ø
C                                                                       RECAØØ6Ø
C  IF THERE IS AN ACTIVE ERROR STATE, THE MESSAGE IS PRINTED            RECAØØ8Ø
C  AND EXECUTION STOPS.                                                 RECAØ1ØØ
C                                                                       RECAØ12Ø
C  IRNEW = Ø LEAVES LRECOV UNCHANGED, WHILE                             RECAØ14Ø
C  IRNEW = 1 GIVES RECOVERY AND                                         RECAØ16Ø
C  IRNEW = 2 TURNS RECOVERY OFF.                                        RECAØ18Ø
C                                                                       RECAØ2ØØ
C  ERROR STATES -                                                       RECAØ22Ø
C                                                                       RECAØ24Ø
C    1 - ILLEGAL VALUE OF IRNEW.                                        RECAØ26Ø
C    2 - CALLED WHILE IN AN ERROR STATE.                                RECAØ28Ø
C                                                                       RECAØ3ØØ
      IF (IRNEW.LT.Ø .OR. IRNEW.GT.2)                                   RECAØ32Ø
     1   CALL SETERR(31HENTSRC - ILLEGAL VALUE OF IRNEW,31,1,2)         RECAØ34Ø
C                                                                       RECAØ36Ø
      IROLD=I8SAVE(2,IRNEW,IRNEW.NE.Ø)                                  RECAØ38Ø
C                                                                       RECAØ4ØØ
C  IF HAVE AN ERROR STATE, STOP EXECUTION.                              RECAØ42Ø
C                                                                       RECAØ44Ø
      IF (I8SAVE(1,Ø,.FALSE.) .NE. Ø) CALL SETERR                       RECAØ46Ø
     1   (39HENTSRC - CALLED WHILE IN AN ERROR STATE,39,2,2)            RECAØ48Ø
C                                                                       RECAØ5ØØ
      RETURN                                                            RECAØ52Ø
C                                                                       RECAØ54Ø
      END                                                               RECAØ56Ø


      SUBROUTINE RETSRC(IROLD)                                          RECBØØØØ
C                                                                       RECBØØ2Ø
C  THIS ROUTINE SETS LRECOV = IROLD.                                    RECBØØ4Ø
C                                                                       RECBØØ6Ø
C  IF THE CURRENT ERROR BECOMES UNRECOVERABLE,                          RECBØØ8Ø
C  THE MESSAGE IS PRINTED AND EXECUTION STOPS.                          RECBØ1ØØ
C                                                                       RECBØ12Ø
C  ERROR STATES -                                                       RECBØ14Ø
C                                                                       RECBØ16Ø
C    1 - ILLEGAL VALUE OF IROLD.                                        RECBØ18Ø
C                                                                       RECBØ2ØØ
      IF (IROLD.LT.1 .OR. IROLD.GT.2)                                   RECBØ22Ø
     1  CALL SETERR(31HRETSRC - ILLEGAL VALUE OF IROLD,31,1,2)          RECBØ24Ø
C                                                                       RECBØ26Ø
      ITEMP=I8SAVE(2,IROLD,.TRUE.)                                      RECBØ28Ø
C                                                                       RECBØ3ØØ
C  IF THE CURRENT ERROR IS NOW UNRECOVERABLE, PRINT AND STOP.           RECBØ32Ø
C                                                                       RECBØ34Ø
```

```
      IF (IROLD.EQ.1 .OR. I8SAVE(1,Ø,.FALSE.).EQ.Ø) RETURN          RECBØ36Ø
C                                                                    RECBØ38Ø
        CALL EPRINT                                                  RECBØ4ØØ
        STOP                                                         RECBØ42Ø
C                                                                    RECBØ44Ø
      END                                                            RECBØ46Ø


      INTEGER FUNCTION NERROR(NERR)                                  ERRNØØØØ
C                                                                    ERRNØØ2Ø
C  RETURNS NERROR = NERR = THE VALUE OF THE ERROR FLAG LERROR.       ERRNØØ4Ø
C                                                                    ERRNØØ6Ø
      NERROR=I8SAVE(1,Ø,.FALSE.)                                     ERRNØØ8Ø
      NERR=NERROR                                                    ERRNØ1ØØ
      RETURN                                                         ERRNØ12Ø
C                                                                    ERRNØ14Ø
      END                                                            ERRNØ16Ø


      SUBROUTINE ERROFF                                              ERRFØØØØ
C                                                                    ERRFØØ2Ø
C  TURNS OFF THE ERROR STATE OFF BY SETTING LERROR=Ø.                ERRFØØ4Ø
C                                                                    ERRFØØ6Ø
      I=I8SAVE(1,Ø,.TRUE.)                                           ERRFØØ8Ø
      RETURN                                                         ERRFØ1ØØ
C                                                                    ERRFØ12Ø
      END                                                            ERRFØ14Ø


      SUBROUTINE EPRINT                                              ERRPØØØØ
C                                                                    ERRPØØ2Ø
C  THIS SUBROUTINE PRINTS THE LAST ERROR MESSAGE, IF ANY.            ERRPØØ4Ø
C                                                                    ERRPØØ6Ø
      INTEGER MESSG(1)                                               ERRPØØ8Ø
C                                                                    ERRPØ1ØØ
      CALL E9RINT(MESSG,1,1,.FALSE.)                                 ERRPØ12Ø
      RETURN                                                         ERRPØ14Ø
C                                                                    ERRPØ16Ø
      END                                                            ERRPØ18Ø


      SUBROUTINE E9RINT(MESSG,NW,NERR,SAVE)                          ERRRØØØØ
C                                                                    ERRRØØ2Ø
C  THIS ROUTINE STORES THE CURRENT ERROR MESSAGE OR PRINTS THE OLD ONE, ERRRØØ4Ø
C  IF ANY, DEPENDING ON WHETHER OR NOT SAVE = .TRUE.                 ERRRØØ6Ø
C                                                                    ERRRØØ8Ø
      INTEGER MESSG(NW)                                              ERRRØ1ØØ
      LOGICAL SAVE                                                   ERRRØ12Ø
C                                                                    ERRRØ14Ø
C  MESSGP STORES AT LEAST THE FIRST 72 CHARACTERS OF THE PREVIOUS    ERRRØ16Ø
C  MESSAGE. ITS LENGTH IS MACHINE DEPENDENT AND MUST BE AT LEAST     ERRRØ18Ø
C                                                                    ERRRØ2ØØ
C       1 + 71/(THE NUMBER OF CHARACTERS STORED PER INTEGER WORD).   ERRRØ22Ø
C                                                                    ERRRØ24Ø
      INTEGER MESSGP(36),FMT(14),CCPLUS                              ERRRØ26Ø
C                                                                    ERRRØ28Ø
C  START WITH NO PREVIOUS MESSAGE.                                   ERRRØ3ØØ
C                                                                    ERRRØ32Ø
      DATA MESSGP(1)/1H1/, NWP/Ø/, NERRP/Ø/                          ERRRØ34Ø
C                                                                    ERRRØ36Ø
C  SET UP THE FORMAT FOR PRINTING THE ERROR MESSAGE.                 ERRRØ38Ø
C  THE FORMAT IS SIMPLY (A1,14X,72AXX) WHERE XX=I1MACH(6) IS THE     ERRRØ4ØØ
C  NUMBER OF CHARACTERS STORED PER INTEGER WORD.                     ERRRØ42Ø
C                                                                    ERRRØ44Ø
      DATA CCPLUS  / 1H+ /                                           ERRRØ46Ø
C                                                                    ERRRØ48Ø
      DATA FMT( 1) / 1H( /                                           ERRRØ5ØØ
      DATA FMT( 2) / 1HA /                                           ERRRØ52Ø
      DATA FMT( 3) / 1H1 /                                           ERRRØ54Ø
      DATA FMT( 4) / 1H, /                                           ERRRØ56Ø
      DATA FMT( 5) / 1H1 /                                           ERRRØ58Ø
      DATA FMT( 6) / 1H4 /                                           ERRRØ6ØØ
      DATA FMT( 7) / 1HX /                                           ERRRØ62Ø
```

```
      DATA FMT( 8) / 1H, /                                              ERRR064Ø
      DATA FMT( 9) / 1H7 /                                              ERRR066Ø
      DATA FMT(1Ø) / 1H2 /                                              ERRR068Ø
      DATA FMT(11) / 1HA /                                              ERRR07ØØ
      DATA FMT(12) / 1HX /                                              ERRR072Ø
      DATA FMT(13) / 1HX /                                              ERRR074Ø
      DATA FMT(14) / 1H) /                                              ERRR076Ø
C                                                                       ERRR078Ø
      IF (.NOT.SAVE) GO TO 2Ø                                           ERRR08ØØ
C                                                                       ERRR082Ø
C  SAVE THE MESSAGE.                                                    ERRR084Ø
C                                                                       ERRR086Ø
        NWP=NW                                                          ERRR088Ø
        NERRP=NERR                                                      ERRR09ØØ
        DO 1Ø I=1,NW                                                    ERRR092Ø
 1Ø     MESSGP(I)=MESSG(I)                                              ERRR094Ø
C                                                                       ERRR096Ø
        GO TO 3Ø                                                        ERRR098Ø
C                                                                       ERRR1ØØØ
 2Ø   IF (I8SAVE(1,Ø,.FALSE.).EQ.Ø) GO TO 3Ø                           ERRR1Ø2Ø
C                                                                       ERRR1Ø4Ø
C  PRINT THE MESSAGE.                                                   ERRR1Ø6Ø
C                                                                       ERRR1Ø8Ø
        IWUNIT=I1MACH(4)                                                ERRR11ØØ
        WRITE(IWUNIT,9ØØØ) NERRP                                        ERRR112Ø
 9ØØØ   FORMAT(7H ERROR ,I4,4H IN )                                     ERRR114Ø
C                                                                       ERRR116Ø
        CALL S88FMT(2,I1MACH(6),FMT(12))                                ERRR118Ø
        WRITE(IWUNIT,FMT) CCPLUS,(MESSGP(I),I=1,NWP)                    ERRR12ØØ
C                                                                       ERRR122Ø
 3Ø   RETURN                                                            ERRR124Ø
C                                                                       ERRR126Ø
      END                                                               ERRR128Ø


      SUBROUTINE S88FMT( N, W, IFMT )                                   ERRMØØØØ
C                                                                       ERRMØØ2Ø
C  S88FMT  REPLACES IFMT(1), ... , IFMT(N) WITH                         ERRMØØ4Ø
C  THE CHARACTERS CORRESPONDING TO THE N LEAST SIGNIFICANT              ERRMØØ6Ø
C  DIGITS OF W.                                                         ERRMØØ8Ø
C                                                                       ERRMØ1ØØ
      INTEGER N,W,IFMT(N)                                               ERRMØ12Ø
C                                                                       ERRMØ14Ø
      INTEGER NT,WT,DIGITS(1Ø)                                          ERRMØ16Ø
C                                                                       ERRMØ18Ø
      DATA DIGITS( 1) / 1HØ /                                           ERRMØ2ØØ
      DATA DIGITS( 2) / 1H1 /                                           ERRMØ22Ø
      DATA DIGITS( 3) / 1H2 /                                           ERRMØ24Ø
      DATA DIGITS( 4) / 1H3 /                                           ERRMØ26Ø
      DATA DIGITS( 5) / 1H4 /                                           ERRMØ28Ø
      DATA DIGITS( 6) / 1H5 /                                           ERRMØ3ØØ
      DATA DIGITS( 7) / 1H6 /                                           ERRMØ32Ø
      DATA DIGITS( 8) / 1H7 /                                           ERRMØ34Ø
      DATA DIGITS( 9) / 1H8 /                                           ERRMØ36Ø
      DATA DIGITS(1Ø) / 1H9 /                                           ERRMØ38Ø
C                                                                       ERRMØ4ØØ
      NT = N                                                            ERRMØ42Ø
      WT = W                                                            ERRMØ44Ø
C                                                                       ERRMØ46Ø
 1Ø   IF (NT .LE. Ø) RETURN                                             ERRMØ48Ø
        IDIGIT = MOD( WT, 1Ø )                                          ERRMØ5ØØ
        IFMT(NT) = DIGITS(IDIGIT+1)                                     ERRMØ52Ø
        WT = WT/1Ø                                                      ERRMØ54Ø
        NT = NT - 1                                                     ERRMØ56Ø
        GO TO 1Ø                                                        ERRMØ58Ø
C                                                                       ERRMØ6ØØ
      END                                                               ERRMØ62Ø


      INTEGER FUNCTION I8SAVE(ISW,IVALUE,SET)                           ERRVØØØØ
C                                                                       ERRVØØ2Ø
C  IF (ISW = 1) I8SAVE RETURNS THE CURRENT ERROR NUMBER AND             ERRVØØ4Ø
C               SETS IT TO IVALUE IF SET = .TRUE. .                     ERRVØØ6Ø
C                                                                       ERRVØØ8Ø
C  IF (ISW = 2) I8SAVE RETURNS THE CURRENT RECOVERY SWITCH AND          ERRVØ1ØØ
C               SETS IT TO IVALUE IF SET = .TRUE. .                     ERRVØ12Ø
```

```
C                                                                       ERRVØ14Ø
      LOGICAL SET                                                       ERRVØ16Ø
C                                                                       ERRVØ18Ø
      INTEGER IPARAM(2)                                                 ERRVØ2ØØ
      EQUIVALENCE (IPARAM(1),LERROR) , (IPARAM(2),LRECOV)               ERRVØ22Ø
C                                                                       ERRVØ24Ø
C  START EXECUTION ERROR FREE AND WITH RECOVERY TURNED OFF.             ERRVØ26Ø
C                                                                       ERRVØ28Ø
      DATA LERROR/Ø/ , LRECOV/2/                                        ERRVØ3ØØ
C                                                                       ERRVØ32Ø
      I8SAVE=IPARAM(ISW)                                                ERRVØ34Ø
      IF (SET) IPARAM(ISW)=IVALUE                                       ERRVØ36Ø
C                                                                       ERRVØ38Ø
      RETURN                                                            ERRVØ4ØØ
C                                                                       ERRVØ42Ø
      END                                                               ERRVØ44Ø


      SUBROUTINE FDUMP                                                  FDMPØØØØ
C  THIS IS A DUMMY ROUTINE TO BE SENT OUT ON                            FDMPØØ2Ø
C  THE PORT SEDIT TAPE                                                  FDMPØØ4Ø
C                                                                       FDMPØØ6Ø
      RETURN                                                            FDMPØØ8Ø
      END                                                               FDMPØ1ØØ
```

Dynamic Storage Allocation Using a Stack

The third package provides a basic mechanism for allocating and deallocating working storage on a storage stack.

In the PORT library [1], for which the package was developed, calls to the general subroutines in the library do not include, in their calling sequences, parameters representing scratch arrays; the work space is allocated and deallocated within the called subprograms.

To implement the stack in Fortran in a portable way, it has been put in labeled COMMON as a double-precision array of length 500:

```
COMMON/CSTAK/DSTAK(500)
DOUBLE PRECISION DSTAK
```

The stack handling capabilities, which are described more fully in [1], include stack allocation and deallocation (releasing space), stack initialization to a size different from the default length of 500 double-precision locations, stack query, i.e. the ability to find out dynamically how much space is available, modification of the length of the latest allocation, and finally, the ability to ascertain certain stack statistics, such as the number of outstanding allocations, the current active length, and the maximum active length achieved.

The various capabilities are summarized as follows:

To allocate (get) N locations of type ITYPE on the stack, set

```
INDEX = ISTKGT(N, ITYPE)
```

which returns an index into the stack for the first of the N items.

To deallocate (release) the last K allocations (not locations but entire allocations):

```
CALL ISTKRL(K)
```

To initialize the stack to, say, 1000 double-precision locations, use the subroutine, ISTKIN(N, ITYPE), as follows:

```
COMMON/CSTAK/DSTAK(1000)
DOUBLE PRECISION DSTAK
                :
  CALL ISTKIN(1000, 4)
```

To find out (query) how much space of type, ITYPE, is left:

NLEFT = ISTKQU(ITYPE)

To modify the length of the current outstanding allocation to N items:

INDEX = ISTKMD(N)

which will modify the length of the allocation to N items and, as in ISTKGT, return the index of the first item of that allocation.

To obtain certain stack statistics use the

INTEGER FUNCTION ISTKST(N)

In summary the user subprograms are:

ISTKGT	allocates space on the stack;
ISTKRL	releases (deallocates) space;
ISTKIN	initializes (sets length) of stack;
ISTKQU	answers query as to space available on stack;
ISTKMD	modifies length of current allocation;
ISTKST	provides statistics on stack usage.

These, in turn, call on the lower-level subprogram:

I0TK00	initializes stack for special cases of nonstandard lengths for INTEGER, REAL, and DOUBLE PRECISION numbers.

ALGORITHM

```
      INTEGER FUNCTION ISTKGT(NITEMS,ITYPE)                              STKE0000
C                                                                        STKE0020
C  ALLOCATES SPACE OUT OF THE INTEGER ARRAY ISTAK (IN COMMON             STKE0040
C  BLOCK CSTAK) FOR AN ARRAY OF LENGTH NITEMS AND OF TYPE                STKE0060
C  DETERMINED BY ITYPE AS FOLLOWS                                        STKE0080
C                                                                        STKE0100
C    1 - LOGICAL                                                         STKE0120
C    2 - INTEGER                                                         STKE0140
C    3 - REAL                                                            STKE0160
C    4 - DOUBLE PRECISION                                                STKE0180
C    5 - COMPLEX                                                         STKE0200
C                                                                        STKE0220
C  ON RETURN, THE ARRAY WILL OCCUPY                                      STKE0240
C                                                                        STKE0260
C    STAK(ISTKGT), STAK(ISTKGT+1), ..., STAK(ISTKGT-NITEMS+1)            STKE0280
C                                                                        STKE0300
C  WHERE STAK IS AN ARRAY OF TYPE ITYPE EQUIVALENCED TO ISTAK.           STKE0320
C                                                                        STKE0340
C  (FOR THOSE WANTING TO MAKE MACHINE DEPENDENT MODIFICATIONS            STKE0360
C  TO SUPPORT OTHER TYPES, CODES 6,7,8,9,10,11 AND 12 HAVE               STKE0380
C  BEEN RESERVED FOR 1/4 LOGICAL, 1/2 LOGICAL, 1/4 INTEGER,              STKE0400
C  1/2 INTEGER, QUAD PRECISION, DOUBLE COMPLEX AND QUAD                  STKE0420
C  COMPLEX, RESPECTIVELY.)                                               STKE0440
C                                                                        STKE0460
C  THE ALLOCATOR RESERVES THE FIRST TEN INTEGER WORDS OF THE STACK       STKE0480
C  FOR ITS OWN INTERNAL BOOK-KEEPING. THESE ARE INITIALIZED BY           STKE0500
C  THE INITIALIZING SUBPROGRAM I0TK00 UPON THE FIRST CALL                STKE0520
C  TO A SUBPROGRAM IN THE ALLOCATION PACKAGE.                            STKE0540
C                                                                        STKE0560
C  THE USE OF THE FIRST FIVE WORDS IS DESCRIBED BELOW.                   STKE0580
C                                                                        STKE0600
C    ISTAK( 1) - LOUT,  THE NUMBER OF CURRENT ALLOCATIONS.               STKE0620
C    ISTAK( 2) - LNOW,  THE CURRENT ACTIVE LENGTH OF THE STACK.          STKE0640
C    ISTAK( 3) - LUSED, THE MAXIMUM VALUE OF ISTAK(2) ACHIEVED.          STKE0660
C    ISTAK( 4) - LMAX,  THE MAXIMUM LENGTH THE STACK.                    STKE0680
C    ISTAK( 5) - LBOOK, THE NUMBER OF WORDS USED FOR BOOKEEPING.         STKE0700
C                                                                        STKE0720
C  THE NEXT FIVE WORDS CONTAIN INTEGERS DESCRIBING THE AMOUNT            STKE0740
C  OF STORAGE ALLOCATED BY THE FORTRAN SYSTEM TO THE VARIOUS             STKE0760
C  DATA TYPES.  THE UNIT OF MEASUREMENT IS ARBITRARY AND MAY             STKE0780
C  BE WORDS, BYTES OR BITS OR WHATEVER IS CONVENIENT.  THE               STKE0800
```

```
C  VALUES CURRENTLY ASSUMED CORRESPOND TO AN ANS FORTRAN                STKEØ82Ø
C  ENVIRONMENT.  FOR SOME MINI-COMPUTER SYSTEMS THE VALUES MAY          STKEØ84Ø
C  HAVE TO BE CHANGED (SEE IØTKØØ).                                     STKEØ86Ø
C                                                                       STKEØ88Ø
C    ISTAK( 6) - THE NUMBER OF UNITS ALLOCATED TO LOGICAL               STKEØ9ØØ
C    ISTAK( 7) - THE NUMBER OF UNITS ALLOCATED TO INTEGER               STKEØ92Ø
C    ISTAK( 8) - THE NUMBER OF UNITS ALLOCATED TO REAL                  STKEØ94Ø
C    ISTAK( 9) - THE NUMBER OF UNITS ALLOCATED TO DOUBLE PRECISION      STKEØ96Ø
C    ISTAK(1Ø) - THE NUMBER OF UNITS ALLOCATED TO COMPLEX               STKEØ98Ø
C                                                                       STKE1ØØØ
C  ERROR STATES -                                                       STKE1Ø2Ø
C                                                                       STKE1Ø4Ø
C    1 - NITEMS .LT. Ø                                                  STKE1Ø6Ø
C    2 - ITYPE .LE. Ø .OR. ITYPE .GE. 6                                 STKE1Ø8Ø
C    3 - LNOW, LUSED, LMAX OR LBOOK OVERWRITTEN                         STKE11ØØ
C    4 - STACK OVERFLOW                                                 STKE112Ø
C                                                                       STKE114Ø
      COMMON /CSTAK/DSTAK                                               STKE116Ø
C                                                                       STKE118Ø
      DOUBLE PRECISION DSTAK(5ØØ)                                       STKE12ØØ
      INTEGER ISTAK(1ØØØ)                                               STKE122Ø
      INTEGER ISIZE(5)                                                  STKE124Ø
C                                                                       STKE126Ø
      LOGICAL INIT                                                      STKE128Ø
C                                                                       STKE13ØØ
      EQUIVALENCE (DSTAK(1),ISTAK(1))                                   STKE132Ø
      EQUIVALENCE (ISTAK(1),LOUT)                                       STKE134Ø
      EQUIVALENCE (ISTAK(2),LNOW)                                       STKE136Ø
      EQUIVALENCE (ISTAK(3),LUSED)                                      STKE138Ø
      EQUIVALENCE (ISTAK(4),LMAX)                                       STKE14ØØ
      EQUIVALENCE (ISTAK(5),LBOOK)                                      STKE142Ø
      EQUIVALENCE (ISTAK(6),ISIZE(1))                                   STKE144Ø
C                                                                       STKE146Ø
      DATA INIT/.TRUE./                                                 STKE148Ø
C                                                                       STKE15ØØ
      IF (INIT) CALL IØTKØØ(INIT,5ØØ,4)                                 STKE152Ø
C                                                                       STKE154Ø
      IF (NITEMS.LT.Ø) CALL SETERR(2ØHISTKGT - NITEMS.LT.Ø,2Ø,1,2)      STKE156Ø
C                                                                       STKE158Ø
      IF (ITYPE.LE.Ø .OR. ITYPE.GE.6) CALL SETERR                       STKE16ØØ
     1   (33HISTKGT - ITYPE.LE.Ø.OR.ITYPE.GE.6,33,2,2)                  STKE162Ø
C                                                                       STKE164Ø
      IF (LNOW.LT.LBOOK.OR.LNOW.GT.LUSED.OR.LUSED.GT.LMAX) CALL SETERR  STKE166Ø
     1   (47HISTKGT - LNOW, LUSED, LMAX OR LBOOK OVERWRITTEN,           STKE168Ø
     2    47,3,2)                                                       STKE17ØØ
C                                                                       STKE172Ø
      ISTKGT = (LNOW*ISIZE(2)-1)/ISIZE(ITYPE) + 2                       STKE174Ø
      I = ( (ISTKGT-1+NITEMS)*ISIZE(ITYPE) - 1 )/ISIZE(2) + 3           STKE176Ø
C                                                                       STKE178Ø
C  STACK OVERFLOW IS AN UNRECOVERABLE ERROR.                            STKE18ØØ
C                                                                       STKE182Ø
      IF (I.GT.LMAX) CALL SETERR(69HISTKGT - STACK TOO SHORT. ENLARGE ITSTKE184Ø
     1 AND CALL ISTKIN IN MAIN PROGRAM.,69,4,2)                         STKE186Ø
C                                                                       STKE188Ø
C  ISTAK(I-1) CONTAINS THE TYPE FOR THIS ALLOCATION.                    STKE19ØØ
C  ISTAK(I  ) CONTAINS A POINTER TO THE END OF THE PREVIOUS·            STKE192Ø
C              ALLOCATION.                                              STKE194Ø
C                                                                       STKE196Ø
      ISTAK(I-1) = ITYPE                                                STKE198Ø
      ISTAK(I  ) = LNOW                                                 STKE2ØØØ
      LOUT = LOUT+1                                                     STKE2Ø2Ø
      LNOW = I                                                          STKE2Ø4Ø
      LUSED = MAXØ(LUSED,LNOW)                                          STKE2Ø6Ø
C                                                                       STKE2Ø8Ø
      RETURN                                                            STKE21ØØ
C                                                                       STKE212Ø
      END                                                               STKE214Ø


      SUBROUTINE ISTKRL(NUMBER)                                         STKCØØØØ
C                                                                       STKCØØ2Ø
C  DE-ALLOCATES THE LAST (NUMBER) ALLOCATIONS MADE IN THE STACK         STKCØØ4Ø
C  BY ISTKGT.                                                           STKCØØ6Ø
C                                                                       STKCØØ8Ø
```

```
C  ERROR STATES -                                                      STKCØ1ØØ
C                                                                      STKCØ12Ø
C    1 - NUMBER .LT. Ø                                                 STKCØ14Ø
C    2 - LNOW, LUSED, LMAX OR LBOOK OVERWRITTEN                        STKCØ16Ø
C    3 - ATTEMPT TO DE-ALLOCATE NON-EXISTENT ALLOCATION                STKCØ18Ø
C    4 - THE POINTER AT ISTAK(LNOW) OVERWRITTEN                        STKCØ2ØØ
C                                                                      STKCØ22Ø
      COMMON /CSTAK/DSTAK                                              STKCØ24Ø
C                                                                      STKCØ26Ø
      DOUBLE PRECISION DSTAK(5ØØ)                                      STKCØ28Ø
      INTEGER ISTAK(1ØØØ)                                              STKCØ3ØØ
      LOGICAL INIT                                                     STKCØ32Ø
C                                                                      STKCØ34Ø
      EQUIVALENCE (DSTAK(1),ISTAK(1))                                  STKCØ36Ø
      EQUIVALENCE (ISTAK(1),LOUT)                                      STKCØ38Ø
      EQUIVALENCE (ISTAK(2),LNOW)                                      STKCØ4ØØ
      EQUIVALENCE (ISTAK(3),LUSED)                                     STKCØ42Ø
      EQUIVALENCE (ISTAK(4),LMAX)                                      STKCØ44Ø
      EQUIVALENCE (ISTAK(5),LBOOK)                                     STKCØ46Ø
C                                                                      STKCØ48Ø
      DATA INIT/.TRUE./                                                STKCØ5ØØ
C                                                                      STKCØ52Ø
      IF (INIT) CALL IØTKØØ(INIT,5ØØ,4)                                STKCØ54Ø
C                                                                      STKCØ56Ø
      IF (NUMBER.LT.Ø) CALL SETERR(2ØHISTKRL - NUMBER.LT.Ø,2Ø,1,2)     STKCØ58Ø
C                                                                      STKCØ6ØØ
      IF (LNOW.LT.LBOOK.OR.LNOW.GT.LUSED.OR.LUSED.GT.LMAX) CALL SETERR STKCØ62Ø
     1   (47HISTKRL - LNOW, LUSED, LMAX OR LBOOK OVERWRITTEN,          STKCØ64Ø
     2    47,2,2)                                                     STKCØ66Ø
C                                                                      STKCØ68Ø
      IN = NUMBER                                                      STKCØ7ØØ
 1Ø     IF (IN.EQ.Ø) RETURN                                           STKCØ72Ø
C                                                                      STKCØ74Ø
        IF (LNOW.LE.LBOOK) CALL SETERR                                 STKCØ76Ø
     1   (55HISTKRL - ATTEMPT TO DE-ALLOCATE NON-EXISTENT ALLOCATION,  STKCØ78Ø
     2    55,3,2)                                                     STKCØ8ØØ
C                                                                      STKCØ82Ø
C     CHECK TO MAKE SURE THE BACK POINTERS ARE MONOTONE.               STKCØ84Ø
C                                                                      STKCØ86Ø
        IF (ISTAK(LNOW).LT.LBOOK.OR.ISTAK(LNOW).GE.LNOW-1) CALL SETERR STKCØ88Ø
     1   (47HISTKRL - THE POINTER AT ISTAK(LNOW) OVERWRITTEN,          STKCØ9ØØ
     2    47,4,2)                                                     STKCØ92Ø
C                                                                      STKCØ94Ø
        LOUT = LOUT-1                                                  STKCØ96Ø
        LNOW = ISTAK(LNOW)                                             STKCØ98Ø
        IN = IN-1                                                      STKC1ØØØ
        GO TO 1Ø                                                       STKC1Ø2Ø
C                                                                      STKC1Ø4Ø
      END                                                              STKC1Ø6Ø


      SUBROUTINE ISTKIN(NITEMS,ITYPE)                                  STKFØØØØ
C                                                                      STKFØØ2Ø
C  INITIALIZES THE STACK ALLOCATOR, SETTING THE LENGTH OF THE STACK.   STKFØØ4Ø
C                                                                      STKFØØ6Ø
C  ERROR STATES -                                                      STKFØØ8Ø
C                                                                      STKFØ1ØØ
C    1 - NITEMS .LE. Ø                                                 STKFØ12Ø
C    2 - ITYPE .LE. Ø .OR. ITYPE .GE. 6                                STKFØ14Ø
C                                                                      STKFØ16Ø
      LOGICAL INIT                                                     STKFØ18Ø
C                                                                      STKFØ2ØØ
      DATA INIT/.TRUE./                                                STKFØ22Ø
C                                                                      STKFØ24Ø
      IF (NITEMS.LE.Ø) CALL SETERR(2ØHISTKIN - NITEMS.LE.Ø,2Ø,1,2)     STKFØ26Ø
C                                                                      STKFØ28Ø
      IF (ITYPE.LE.Ø.OR.ITYPE.GE.6) CALL SETERR                        STKFØ3ØØ
     1   (33HISTKIN - ITYPE.LE.Ø.OR.ITYPE.GE.6,33,2,2)                 STKFØ32Ø
C                                                                      STKFØ34Ø
      IF (INIT) CALL IØTKØØ(INIT,NITEMS,ITYPE)                         STKFØ36Ø
C                                                                      STKFØ38Ø
      RETURN                                                           STKFØ4ØØ
C                                                                      STKFØ42Ø
      END                                                              STKFØ44Ø
```

```
      INTEGER FUNCTION ISTKQU(ITYPE)                                    STKAØØØØ
C                                                                       STKAØØ2Ø
C  RETURNS THE NUMBER OF ITEMS OF TYPE ITYPE THAT REMAIN                STKAØØ4Ø
C  TO BE ALLOCATED IN ONE REQUEST.                                      STKAØØ6Ø
C                                                                       STKAØØ8Ø
C  ERROR STATES -                                                       STKAØ1ØØ
C                                                                       STKAØ12Ø
C    1 - LNOW, LUSED, LMAX OR LBOOK OVERWRITTEN                         STKAØ14Ø
C    2 - ITYPE .LE. Ø .OR. ITYPE .GE. 6                                 STKAØ16Ø
C                                                                       STKAØ18Ø
      COMMON /CSTAK/DSTAK                                               STKAØ2ØØ
C                                                                       STKAØ22Ø
      DOUBLE PRECISION DSTAK(5ØØ)                                       STKAØ24Ø
      INTEGER ISTAK(1ØØØ)                                               STKAØ26Ø
      INTEGER ISIZE(5)                                                  STKAØ28Ø
C                                                                       STKAØ3ØØ
      LOGICAL INIT                                                      STKAØ32Ø
C                                                                       STKAØ34Ø
      EQUIVALENCE (DSTAK(1),ISTAK(1))                                   STKAØ36Ø
      EQUIVALENCE (ISTAK(2),LNOW)                                       STKAØ38Ø
      EQUIVALENCE (ISTAK(3),LUSED)                                      STKAØ4ØØ
      EQUIVALENCE (ISTAK(4),LMAX)                                       STKAØ42Ø
      EQUIVALENCE (ISTAK(5),LBOOK)                                      STKAØ44Ø
      EQUIVALENCE (ISTAK(6),ISIZE(1))                                   STKAØ46Ø
C                                                                       STKAØ48Ø
      DATA INIT/.TRUE./                                                 STKAØ5ØØ
C                                                                       STKAØ52Ø
      IF (INIT) CALL IØTKØØ(INIT,5ØØ,4)                                 STKAØ54Ø
C                                                                       STKAØ56Ø
      IF (LNOW.LT.LBOOK.OR.LNOW.GT.LUSED.OR.LUSED.GT.LMAX) CALL SETERR  STKAØ58Ø
     1   (47HISTKQU - LNOW, LUSED, LMAX OR LBOOK OVERWRITTEN,           STKAØ6ØØ
     2    47,1,2)                                                      STKAØ62Ø
C                                                                       STKAØ64Ø
      IF (ITYPE.LE.Ø.OR.ITYPE.GE.6) CALL SETERR                         STKAØ66Ø
     1   (33HISTKQU - ITYPE.LE.Ø.OR.ITYPE.GE.6,33,2,2)                  STKAØ68Ø
C                                                                       STKAØ7ØØ
      ISTKQU = MAXØ( ((LMAX-2)*ISIZE(2))/ISIZE(ITYPE)                   STKAØ72Ø
     1             - (LNOW*ISIZE(2)-1)/ISIZE(ITYPE)                     STKAØ74Ø
     2             - 1, Ø )                                             STKAØ76Ø
C                                                                       STKAØ78Ø
      RETURN                                                            STKAØ8ØØ
C                                                                       STKAØ82Ø
      END                                                               STKAØ84Ø


      INTEGER FUNCTION ISTKMD(NITEMS)                                   STKBØØØØ
C                                                                       STKBØØ2Ø
C  CHANGES THE LENGTH OF THE FRAME AT THE TOP OF THE STACK              STKBØØ4Ø
C  TO NITEMS.                                                           STKBØØ6Ø
C                                                                       STKBØØ8Ø
C  ERROR STATES -                                                       STKBØ1ØØ
C                                                                       STKBØ12Ø
C    1 - LNOW OVERWRITTEN                                               STKBØ14Ø
C    2 - ISTAK(LNOWO-1) OVERWRITTEN                                     STKBØ16Ø
C                                                                       STKBØ18Ø
      COMMON /CSTAK/DSTAK                                               STKBØ2ØØ
C                                                                       STKBØ22Ø
      DOUBLE PRECISION DSTAK(5ØØ)                                       STKBØ24Ø
        TEGER ISTAK(1ØØØ)                                               STKBØ26Ø
C                                                                       STKBØ28Ø
      EQUIVALENCE (DSTAK(1),ISTAK(1))                                   STKBØ3ØØ
      EQUIVALENCE (ISTAK(2),LNOW)                                       STKBØ32Ø
C                                                                       STKBØ34Ø
      LNOWO = LNOW                                                      STKBØ36Ø
      CALL ISTKRL(1)                                                    STKBØ38Ø
C                                                                       STKBØ4ØØ
      ITYPE = ISTAK(LNOWO-1)                                            STKBØ42Ø
C                                                                       STKBØ44Ø
      IF (ITYPE.LE.Ø.OR.ITYPE.GE.6) CALL SETERR                         STKBØ46Ø
     1   (35HISTKMD - ISTAK(LNOWO-1) OVERWRITTEN,35,1,2)                STKBØ48Ø
C                                                                       STKBØ5ØØ
      ISTKMD = ISTKGT(NITEMS,ITYPE)                                     STKBØ52Ø
C                                                                       STKBØ54Ø
      RETURN                                                            STKBØ56Ø
C                                                                       STKBØ58Ø
      END                                                               STKBØ6ØØ
```

```
      INTEGER FUNCTION ISTKST(NFACT)                                    STKSØØØØ
C                                                                       STKSØØ2Ø
C  RETURNS CONTROL INFORMATION AS FOLLOWS                               STKSØØ4Ø
C                                                                       STKSØØ6Ø
C  NFACT    ITEM RETURNED                                               STKSØØ8Ø
C                                                                       STKSØ1ØØ
C    1         LOUT,  THE NUMBER OF CURRENT ALLOCATIONS                 STKSØ12Ø
C    2         LNOW,  THE CURRENT ACTIVE LENGTH                         STKSØ14Ø
C    3         LUSED, THE MAXIMUM USED                                  STKSØ16Ø
C    4         LMAX,  THE MAXIMUM ALLOWED                               STKSØ18Ø
C                                                                       STKSØ2ØØ
      COMMON /CSTAK/DSTAK                                               STKSØ22Ø
C                                                                       STKSØ24Ø
      DOUBLE PRECISION DSTAK(5ØØ)                                       STKSØ26Ø
      INTEGER ISTAK(1ØØØ)                                               STKSØ28Ø
      INTEGER ISTATS(4)                                                 STKSØ3ØØ
      LOGICAL INIT                                                      STKSØ32Ø
C                                                                       STKSØ34Ø
      EQUIVALENCE (DSTAK(1),ISTAK(1))                                   STKSØ36Ø
      EQUIVALENCE (ISTAK(1),ISTATS(1))                                  STKSØ38Ø
C                                                                       STKSØ4ØØ
      DATA INIT/.TRUE./                                                 STKSØ42Ø
C                                                                       STKSØ44Ø
      IF (INIT) CALL IØTKØØ(INIT,5ØØ,4)                                 STKSØ46Ø
C                                                                       STKSØ48Ø
      IF (NFACT.LE.Ø.OR.NFACT.GE.5) CALL SETERR                         STKSØ5ØØ
     1   (33HISTKST - NFACT.LE.Ø.OR.NFACT.GE.5,33,1,2)                  STKSØ52Ø
C                                                                       STKSØ54Ø
      ISTKST = ISTATS(NFACT)                                            STKSØ56Ø
C                                                                       STKSØ58Ø
      RETURN                                                            STKSØ6ØØ
C                                                                       STKSØ62Ø
      END                                                               STKSØ64Ø


      SUBROUTINE IØTKØØ(LARG,NITEMS,ITYPE)                              STKGØØØØ
C                                                                       STKGØØ2Ø
C  INITIALIZES THE STACK TO NITEMS OF TYPE ITYPE                        STKGØØ4Ø
C                                                                       STKGØØ6Ø
      COMMON /CSTAK/DSTAK                                               STKGØØ8Ø
C                                                                       STKGØ1ØØ
      DOUBLE PRECISION DSTAK(5ØØ)                                       STKGØ12Ø
      INTEGER ISTAK(1ØØØ)                                               STKGØ14Ø
      LOGICAL LARG,INIT                                                 STKGØ16Ø
      INTEGER ISIZE(5)                                                  STKGØ18Ø
C                                                                       STKGØ2ØØ
      EQUIVALENCE (DSTAK(1),ISTAK(1))                                   STKGØ22Ø
      EQUIVALENCE (ISTAK(1),LOUT)                                       STKGØ24Ø
      EQUIVALENCE (ISTAK(2),LNOW)                                       STKGØ26Ø
      EQUIVALENCE (ISTAK(3),LUSED)                                      STKGØ28Ø
      EQUIVALENCE (ISTAK(4),LMAX)                                       STKGØ3ØØ
      EQUIVALENCE (ISTAK(5),LBOOK)                                      STKGØ32Ø
      EQUIVALENCE (ISTAK(6),ISIZE(1))                                   STKGØ34Ø
C                                                                       STKGØ36Ø
      DATA INIT/.FALSE./                                                STKGØ38Ø
C                                                                       STKGØ4ØØ
      LARG = .FALSE.                                                    STKGØ42Ø
      IF (INIT) RETURN                                                  STKGØ44Ø
C                                                                       STKGØ46Ø
C  HERE TO INITIALIZE                                                   STKGØ48Ø
C                                                                       STKGØ5ØØ
      INIT = .TRUE.                                                     STKGØ52Ø
C                                                                       STKGØ54Ø
C  SET DATA SIZES APPROPRIATE FOR A STANDARD CONFORMING                 STKGØ56Ø
C  FORTRAN SYSTEM USING THE FORTRAN "STORAGE UNIT" AS THE               STKGØ58Ø
C  MEASURE OF SIZE.                                                     STKGØ6ØØ
C                                                                       STKGØ62Ø
C  LOGICAL                                                              STKGØ64Ø
      ISIZE(1) = 1                                                      STKGØ66Ø
C  INTEGER                                                              STKGØ68Ø
      ISIZE(2) = 1                                                      STKGØ7ØØ
C  REAL                                                                 STKGØ72Ø
      ISIZE(3) = 1                                                      STKGØ74Ø
C  DOUBLE PRECISION                                                     STKGØ76Ø
      ISIZE(4) = 2                                                      STKGØ78Ø
```

```
C  COMPLEX                                                          STKGØ8ØØ
      ISIZE(5) = 2                                                  STKGØ82Ø
C                                                                   STKGØ84Ø
      LBOOK = 1Ø                                                    STKGØ86Ø
      LNOW  = LBOOK                                                 STKGØ88Ø
      LUSED = LBOOK                                                 STKGØ9ØØ
      LMAX  = MAXØ( (NITEMS*ISIZE(ITYPE))/ISIZE(2), 12 )            STKGØ92Ø
      LOUT  = Ø                                                     STKGØ94Ø
C                                                                   STKGØ96Ø
      RETURN                                                        STKGØ98Ø
C                                                                   STKG1ØØØ
      END                                                           STKG1Ø2Ø
```

ACM Transactions on Mathematical Software, Vol. 5, No. 4, December 1979, Page 524.

REMARK ON ALGORITHM 528

Framework for a Portable Library [Z]
[P.A. Fox, A.D. Hall, and N.L. Schryer, *ACM Trans. Math. Software 4*, 2 (June 1978), 177–188]

Phyllis Fox [Recd 11 September 1979]
Bell Laboratories, 600 Mountain Ave., Murray Hill, NJ 07974

In the machine-dependent constants for the PDP 10 computer with KI processor, a transcription error has been found in the double-precision value for $\log_{10} b$.

The DATA line should read as follows:

```
C  DATA LOG10(1),LOG10(2)/"177464202324, "047674776746/          MCHD1680
```

(See the complete listing of Algorithm 528, available from the ACM Algorithms Distribution Service or to be found in "Collected Algorithms from ACM.")

ACKNOWLEDGMENT

The author is grateful to Prof. Nelson Beebe of the Department of Physics, University of Utah, for detecting the incorrect value.

ALGORITHM 529
Permutations to Block Triangular Form [F1]

I. S. DUFF and J. K. REID
AERE Harwell

Key Words and Phrases: symmetric permutations, block triangular form, depth first search algorithm, sparse matrices
CR Categories: 5.0, 5.1, 5.3, 5.4
Language: ANSI Fortran

DESCRIPTION

Given the column numbers of the nonzeros in each row of a sparse matrix, this subroutine finds a symmetric permutation that makes the matrix block lower triangular. It can also be interpreted as accepting the row numbers of the nonzeros in each column and symmetrically permuting to block upper triangular form. If the user submits a matrix with zeros on the diagonal, subroutine MC13D might give a block triangular form which could be further reduced by unsymmetric permutations. To obtain the best results, the user is advised first to permute the matrix so that it has a zero-free diagonal. This can be done by Harwell subroutine MC21A (Duff [1]).

The subroutine is evoked by the Fortran statement

```
CALL MC13D(N,ICN,LICN,IP,LENR,IOR,IB,NUM,IW)
```

where the parameters are described in the listing given here.

The subroutine is based on Tarjan's depth first search algorithm [3], and its design is described in detail in Duff and Reid [2]. They also discuss experimental results from using this subroutine which has been tested on a wide range of both structured and randomly generated matrices.

This routine has been written in ANSI Fortran. Special comment cards have been included so that Harwell subroutine OEØ4A can be used to make an IBM Fortran version that uses half-length integers (INTEGER*2) for all the arrays except IP. This approximately halves the core requirements at the cost of restricting the order of the system to $2^{15} - 1$.

REFERENCES

1. DUFF, I.S. On algorithms for obtaining a maximum transversal. To appear as a Harwell Report.
2. DUFF, I.S., AND REID, J.K. An implementation of Tarjan's algorithm for the block triangularization of a matrix. *ACM Trans. Math. Software 4*, 2 (June 1978), 137–147.
3. TARJAN, R.E. Depth first search and linear graph algorithms. *SIAM J. Comput. 1*, pp. 146–160.

Received 8 April 1976; revised 20 August 1976.

Authors' address: Computer Science and Systems Division, Building 8.9, AERE Harwell, Didcot, Oxfordshire, OX11 ORA, England.

ALGORITHM

[Subroutines MC13D and MC13E are printed here. A test deck for MC13D is available from the ACM Distribution Service (see inside back cover for order form), or may be found in "Collected Algorithms from ACM."]

```
C I IS THE IBM VERSION                                                  SI/MC1   1Ø
C S IS THE STANDARD FORTRAN VERSION                                       MC1   2Ø
C                                                                         MC1   3Ø
      SUBROUTINE MC13D(N, ICN, LICN, IP, LENR, IOR, IB, NUM, IW)          MC1   4Ø
C                                                                         MC1   5Ø
C DESCRIPTION OF PARAMETERS.                                              MC1   6Ø
C INPUT VARIABLES  .... N,ICN,LICN,IP,LENR.                               MC1   7Ø
C OUTPUT VARIABLES  IOR,IB,NUM.                                           MC1   8Ø
C                                                                         MC1   9Ø
C N   ORDER OF THE MATRIX.                                                MC1  1ØØ
C ICN ARRAY CONTAINING THE COLUMN INDICES OF THE NON-ZEROS.  THOSE        MC1  11Ø
C     BELONGING TO A SINGLE ROW MUST BE CONTIGUOUS BUT THE ORDERING       MC1  12Ø
C     OF COLUMN INDICES WITHIN EACH ROW IS UNIMPORTANT AND WASTED         MC1  13Ø
C     SPACE BETWEEN ROWS IS PERMITTED.                                    MC1  14Ø
C LICN  LENGTH OF ARRAY ICN.                                              MC1  15Ø
C IP  IP(I), I=1,2,...N, IS THE POSITION IN ARRAY ICN OF THE FIRST        MC1  16Ø
C     COLUMN INDEX OF A NON-ZERO IN ROW I.                                MC1  17Ø
C LENR  LENR(I) IS THE NUMBER OF NON-ZEROS IN ROW I, I=1,2,...N.          MC1  18Ø
C IOR  IOR(I) GIVES THE POSITION IN THE ORIGINAL ORDERING OF THE ROW      MC1  19Ø
C     OR COLUMN WHICH IS IN POSITION I IN THE PERMUTED FORM, I=1,2,..N.   MC1  2ØØ
C IB  IB(I) IS THE ROW NUMBER IN THE PERMUTED MATRIX OF THE BEGINNING     MC1  21Ø
C     OF BLOCK I, I=1,2,...NUM.                                           MC1  22Ø
C NUM NUMBER OF BLOCKS FOUND.                                             MC1  23Ø
C IW  WORK ARRAY .. SEE LATER COMMENTS.                                   MC1  24Ø
C                                                                         MC1  25Ø
      INTEGER IP(N)                                                       MC1  26Ø
C     INTEGER*2 ICN(LICN),LENR(N),IOR(N),IB(N),IW(N,3)                 I/MC1  27Ø
      INTEGER ICN(LICN), LENR(N), IOR(N), IB(N), IW(N,3)                  MC1  28Ø
      CALL MC13E(N, ICN, LICN, IP, LENR, IOR, IB, NUM, IW(1,1),           MC1  29Ø
     *  IW(1,2), IW(1,3))                                                 MC1  3ØØ
      RETURN                                                              MC1  31Ø
      END                                                                 MC1  32Ø

      SUBROUTINE MC13E(N, ICN, LICN, IP, LENR, ARP, IB, NUM, LOWL,        MC1   1Ø
     *  NUMB, PREV)                                                       MC1   2Ø
      INTEGER STP, DUMMY                                                  MC1   3Ø
      INTEGER IP(N)                                                       MC1   4Ø
C                                                                         MC1   5Ø
C ARP(I) IS ONE LESS THAN THE NUMBER OF UNSEARCHED EDGES LEAVING          MC1   6Ø
C     NODE I.  AT THE END OF THE ALGORITHM IT IS SET TO A                 MC1   7Ø
C     PERMUTATION WHICH PUTS THE MATRIX IN BLOCK LOWER                    MC1   8Ø
C     TRIANGULAR FORM.                                                    MC1   9Ø
C IB(I) IS THE POSITION IN THE ORDERING OF THE START OF THE ITH           MC1  1ØØ
C     BLOCK.  IB(N+1-I) HOLDS THE NODE NUMBER OF THE ITH NODE             MC1  11Ø
C     ON THE STACK.                                                       MC1  12Ø
C LOWL(I) IS THE SMALLEST STACK POSITION OF ANY NODE TO WHICH A PATH      MC1  13Ø
C     FROM NODE I HAS BEEN FOUND.  IT IS SET TO N+1 WHEN NODE I           MC1  14Ø
C     IS REMOVED FROM THE STACK.                                          MC1  15Ø
C NUMB(I) IS THE POSITION OF NODE I IN THE STACK IF IT IS ON              MC1  16Ø
C     IT, IS THE PERMUTED ORDER OF NODE I FOR THOSE NODES                 MC1  17Ø
C     WHOSE FINAL POSITION HAS BEEN FOUND AND IS OTHERWISE ZERO.          MC1  18Ø
C PREV(I) IS THE NODE AT THE END OF THE PATH WHEN NODE I WAS              MC1  19Ø
C     PLACED ON THE STACK.                                                MC1  2ØØ
C     INTEGER*2 ICN(LICN),LENR(N),ARP(N),IB(N),LOWL(N),NUMB(N),        I/MC1  21Ø
C    1PREV(N)                                                          I/MC1  22Ø
      INTEGER ICN(LICN), LENR(N), ARP(N), IB(N), LOWL(N), NUMB(N),        MC1  23Ø
     *  PREV(N)                                                           MC1  24Ø
C                                                                         MC1  25Ø
C                                                                         MC1  26Ø
C   ICNT IS THE NUMBER OF NODES WHOSE POSITIONS IN FINAL ORDERING HAVE    MC1  27Ø
C     BEEN FOUND.                                                         MC1  28Ø
      ICNT = Ø                                                            MC1  29Ø
C NUM IS THE NUMBER OF BLOCKS THAT HAVE BEEN FOUND.                       MC1  3ØØ
      NUM = Ø                                                             MC1  31Ø
      NNM1 = N + N - 1                                                    MC1  32Ø
C                                                                         MC1  33Ø
C INITIALIZATION OF ARRAYS.                                               MC1  34Ø
      DO 1Ø J=1,N                                                         MC1  35Ø
        NUMB(J) = Ø                                                       MC1  36Ø
```

```
      ARP(J) = LENR(J) - 1                                                    MC1  37Ø
   1Ø CONTINUE                                                                MC1  38Ø
C                                                                             MC1  39Ø
C                                                                             MC1  4ØØ
      DO 9Ø ISN=1,N                                                           MC1  41Ø
C LOOK FOR A STARTING NODE                                                    MC1  42Ø
      IF (NUMB(ISN).NE.Ø) GO TO 9Ø                                            MC1  43Ø
      IV = ISN                                                                MC1  44Ø
C IST IS THE NUMBER OF NODES ON THE STACK ... IT IS THE STACK POINTER.        MC1  45Ø
      IST = 1                                                                 MC1  46Ø
C PUT NODE IV AT BEGINNING OF STACK.                                          MC1  47Ø
      LOWL(IV) = 1                                                            MC1  48Ø
      NUMB(IV) = 1                                                            MC1  49Ø
      IB(N) = IV                                                              MC1  5ØØ
C                                                                             MC1  51Ø
C THE BODY OF THIS LOOP PUTS A NEW NODE ON THE STACK OR BACKTRACKS.           MC1  52Ø
      DO 8Ø DUMMY=1,NNM1                                                      MC1  53Ø
        I1 = ARP(IV)                                                          MC1  54Ø
C HAVE ALL EDGES LEAVING NODE IV BEEN SEARCHED.                               MC1  55Ø
        IF (I1.LT.Ø) GO TO 3Ø                                                 MC1  56Ø
        I2 = IP(IV) + LENR(IV) - 1                                            MC1  57Ø
        I1 = I2 - I1                                                          MC1  58Ø
C                                                                             MC1  59Ø
C LOOK AT EDGES LEAVING NODE IV UNTIL ONE ENTERS A NEW NODE OR                MC1  6ØØ
C     ALL EDGES ARE EXHAUSTED.                                                MC1  61Ø
        DO 2Ø II=I1,I2                                                        MC1  62Ø
          IW = ICN(II)                                                        MC1  63Ø
C HAS NODE IW BEEN ON STACK ALREADY.                                          MC1  64Ø
          IF (NUMB(IW).EQ.Ø) GO TO 7Ø                                         MC1  65Ø
C UPDATE VALUE OF LOWL(IV) IF NECESSARY.                                      MC1  66Ø
          IF (LOWL(IW).LT.LOWL(IV)) LOWL(IV) = LOWL(IW)                       MC1  67Ø
   2Ø   CONTINUE                                                              MC1  68Ø
C                                                                             MC1  69Ø
C THERE ARE NO MORE EDGES LEAVING NODE IV.                                    MC1  7ØØ
        ARP(IV) = -1                                                          MC1  71Ø
C IS NODE IV THE ROOT OF A BLOCK.                                             MC1  72Ø
   3Ø   IF (LOWL(IV).LT.NUMB(IV)) GO TO 6Ø                                    MC1  73Ø
C                                                                             MC1  74Ø
C ORDER NODES IN A BLOCK.                                                     MC1  75Ø
        NUM = NUM + 1                                                         MC1  76Ø
        IST1 = N + 1 - IST                                                    MC1  77Ø
        LCNT = ICNT + 1                                                       MC1  78Ø
C PEEL BLOCK OFF THE TOP OF THE STACK STARTING AT THE TOP AND                 MC1  79Ø
C     WORKING DOWN TO THE ROOT OF THE BLOCK.                                  MC1  8ØØ
        DO 4Ø STP=IST1,N                                                      MC1  81Ø
          IW = IB(STP)                                                        MC1  82Ø
          LOWL(IW) = N + 1                                                    MC1  83Ø
          ICNT = ICNT + 1                                                     MC1  84Ø
          NUMB(IW) = ICNT                                                     MC1  85Ø
          IF (IW.EQ.IV) GO TO 5Ø                                              MC1  86Ø
   4Ø   CONTINUE                                                              MC1  87Ø
   5Ø   IST = N - STP                                                         MC1  88Ø
        IB(NUM) = LCNT                                                        MC1  89Ø
C ARE THERE ANY NODES LEFT ON THE STACK.                                      MC1  9ØØ
        IF (IST.NE.Ø) GO TO 6Ø                                                MC1  91Ø
C HAVE ALL THE NODES BEEN ORDERED.                                            MC1  92Ø
        IF (ICNT.LT.N) GO TO 9Ø                                               MC1  93Ø
        GO TO 1ØØ                                                             MC1  94Ø
C                                                                             MC1  95Ø
C BACKTRACK TO PREVIOUS NODE ON PATH.                                         MC1  96Ø
   6Ø   IW = IV                                                               MC1  97Ø
        IV = PREV(IV)                                                         MC1  98Ø
C UPDATE VALUE OF LOWL(IV) IF NECESSARY.                                      MC1  99Ø
        IF (LOWL(IW).LT.LOWL(IV)) LOWL(IV) = LOWL(IW)                         MC1 1ØØØ
        GO TO 8Ø                                                              MC1 1Ø1Ø
C                                                                             MC1 1Ø2Ø
C PUT NEW NODE ON THE STACK.                                                  MC1 1Ø3Ø
   7Ø   ARP(IV) = I2 - II - 1                                                 MC1 1Ø4Ø
        PREV(IW) = IV                                                         MC1 1Ø5Ø
        IV = IW                                                               MC1 1Ø6Ø
        IST = IST + 1                                                         MC1 1Ø7Ø
        LOWL(IV) = IST                                                        MC1 1Ø8Ø
        NUMB(IV) = IST                                                        MC1 1Ø9Ø
        K = N + 1 - IST                                                       MC1 11ØØ
        IB(K) = IV                                                            MC1 111Ø
   8Ø CONTINUE                                                                MC1 112Ø
```

```
C                                                                       MC1 113Ø
   9Ø CONTINUE                                                          MC1 114Ø
C                                                                       MC1 115Ø
C                                                                       MC1 116Ø
C PUT PERMUTATION IN THE REQUIRED FORM.                                 MC1 117Ø
  1ØØ DO 11Ø I=1,N                                                      MC1 118Ø
        II = NUMB(I)                                                    MC1 119Ø
        ARP(II) = I                                                     MC1 12ØØ
  11Ø CONTINUE                                                          MC1 121Ø
      RETURN                                                            MC1 122Ø
      END                                                               MC1 123Ø


C TEST DECK FOR MC13D ... RUNS ON RANDOM MATRICES ...                   MAN   1Ø
      INTEGER IP(5Ø), ICN(1ØØØ), IOR(5Ø), IB(51), IW(15Ø), LENR(5Ø)     MAN   2Ø
      INTEGER BLANK, EX, HOLD(1ØØ)                                      MAN   3Ø
      LOGICAL A(5Ø,5Ø)                                                  MAN   4Ø
      DATA BLANK, EX, NOT /1H ,1HX,1HØ/                                 MAN   5Ø
      MM = 5Ø                                                           MAN   6Ø
      LICN = 1ØØØ                                                       MAN   7Ø
C                                                                       MAN   8Ø
C MAIN LOOP.                                                            MAN   9Ø
   1Ø READ (5,99999) N, IPP                                             MAN  1ØØ
      IF (N.EQ.Ø) GO TO 1ØØ                                             MAN  11Ø
      WRITE (6,99998) N, IPP                                            MAN  12Ø
      DO 3Ø J=1,N                                                       MAN  13Ø
        DO 2Ø I=1,N                                                     MAN  14Ø
          A(I,J) = .FALSE.                                              MAN  15Ø
   2Ø   CONTINUE                                                        MAN  16Ø
        A(J,J) = .TRUE.                                                 MAN  17Ø
   3Ø CONTINUE                                                          MAN  18Ø
      IF (IPP.EQ.Ø) GO TO 6Ø                                            MAN  19Ø
      DO 5Ø K9=1,IPP                                                    MAN  2ØØ
C THESE STATEMENTS SHOULD BE REPLACED BY CALLS TO YOUR FAVOURITE        MAN  21Ø
C     RANDOM NUMBER GENERATOR TO PLACE TWO PSEUDO-RANDOM NUMBERS        MAN  22Ø
C     BETWEEN 1 AND N IN THE VARIABLES I AND J.                         MAN  23Ø
   4Ø   CALL FAØ1BS(N, I)                                               MAN  24Ø
        CALL FAØ1BS(N, J)                                               MAN  25Ø
        IF (A(I,J)) GO TO 4Ø                                            MAN  26Ø
        A(I,J) = .TRUE.                                                 MAN  27Ø
   5Ø CONTINUE                                                          MAN  28Ø
C SETUP CONVERTS MATRIX A(I,J) TO REQUIRED SPARSITY-ORIENTED            MAN  29Ø
C     STORAGE FORMAT.                                                   MAN  3ØØ
   6Ø CALL SETUP(N, A, MM, IP, ICN, LICN, LENR)                         MAN  31Ø
      CALL MC13D(N, ICN, LICN, IP, LENR, IOR, IB, NUM, IW)              MAN  32Ø
C OUTPUT REORDERED MATRIX WITH BLOCKING TO IMPROVE CLARITY.             MAN  33Ø
      IF (NUM.EQ.1) WRITE (6,99997) NUM                                 MAN  34Ø
      IF (NUM.NE.1) WRITE (6,99996) NUM                                 MAN  35Ø
      IB(NUM+1) = N + 1                                                 MAN  36Ø
      INDEX = 1ØØ                                                       MAN  37Ø
      IBLOCK = 1                                                        MAN  38Ø
      DO 9Ø I=1,N                                                       MAN  39Ø
        DO 7Ø IJ=1,INDEX                                                MAN  4ØØ
          HOLD(IJ) = BLANK                                              MAN  41Ø
   7Ø   CONTINUE                                                        MAN  42Ø
        IF (I.EQ.IB(IBLOCK)) WRITE (6,99995)                            MAN  43Ø
        IF (I.EQ.IB(IBLOCK)) IBLOCK = IBLOCK + 1                        MAN  44Ø
        JBLOCK = 1                                                      MAN  45Ø
        INDEX = Ø                                                       MAN  46Ø
        DO 8Ø J=1,N                                                     MAN  47Ø
          IF (J.EQ.IB(JBLOCK)) INDEX = INDEX + 1                        MAN  48Ø
          IF (J.EQ.IB(JBLOCK)) HOLD(INDEX) = BLANK                      MAN  49Ø
          IF (J.EQ.IB(JBLOCK)) JBLOCK = JBLOCK + 1                      MAN  5ØØ
          INDEX = INDEX + 1                                             MAN  51Ø
          II = IOR(I)                                                   MAN  52Ø
          JJ = IOR(J)                                                   MAN  53Ø
          IF (A(II,JJ)) HOLD(INDEX) = EX                                MAN  54Ø
          IF (.NOT.A(II,JJ)) HOLD(INDEX) = NOT                          MAN  55Ø
   8Ø   CONTINUE                                                        MAN  56Ø
        WRITE (6,99994) HOLD                                            MAN  57Ø
   9Ø CONTINUE                                                          MAN  58Ø
      WRITE (6,99993) (IB(I),I=1,NUM)                                   MAN  59Ø
      GO TO 1Ø                                                          MAN  6ØØ
C                                                                       MAN  61Ø
C FORMAT STATEMENTS.                                                    MAN  62Ø
  1ØØ STOP                                                              MAN  63Ø
```

```
99999 FORMAT (2I4)                                                    MAN  64Ø
99998 FORMAT (1H1, 2ØH MATRIX IS OF ORDER , I3, 9H AND HAS , I3,      MAN  65Ø
     *  23H OFF-DIAGONAL NON-ZEROS)                                   MAN  66Ø
99997 FORMAT (///31H THE REORDERED MATRIX WHICH HAS, I3, 1ØH BLOCK IS , MAN  67Ø
     *  11HOF THE FORM/)                                              MAN  68Ø
99996 FORMAT (///31H THE REORDERED MATRIX WHICH HAS, I3, 1ØH BLOCKS IS, MAN  69Ø
     *  12H OF THE FORM/)                                             MAN  7ØØ
99995 FORMAT (3X)                                                     MAN  71Ø
99994 FORMAT (1X, 1ØØA1)                                              MAN  72Ø
99993 FORMAT (/46H THE STARTING POINT FOR EACH BLOCK IS GIVEN BY//     MAN  73Ø
     *  2Ø(2X, I4))                                                   MAN  74Ø
      END                                                             MAN  75Ø



      SUBROUTINE SETUP(N, A, MM, IP, ICN, LICN, LENR)                 SET   1Ø
      LOGICAL A(MM,MM)                                                SET   2Ø
      INTEGER IP(N), ICN(LICN), LENR(N)                               SET   3Ø
      DO 1Ø I=1,N                                                     SET   4Ø
        LENR(I) = Ø                                                   SET   5Ø
   1Ø CONTINUE                                                        SET   6Ø
      IND = 1                                                         SET   7Ø
      DO 3Ø I=1,N                                                     SET   8Ø
        IP(I) = IND                                                   SET   9Ø
        DO 2Ø J=1,N                                                   SET  1ØØ
          IF (.NOT.A(I,J)) GO TO 2Ø                                   SET  11Ø
          LENR(I) = LENR(I) + 1                                       SET  12Ø
          ICN(IND) = J                                                SET  13Ø
          IND = IND + 1                                               SET  14Ø
   2Ø   CONTINUE                                                      SET  15Ø
   3Ø CONTINUE                                                        SET  16Ø
      RETURN                                                          SET  17Ø
      END                                                             SET  18Ø
```

ALGORITHM 530
An Algorithm for Computing the Eigensystem of Skew-Symmetric Matrices and a Class of Symmetric Matrices [F2]

R. C. WARD and L. J. GRAY
Union Carbide Corporation, Nuclear Division

Key Words and Phrases: eigenvalues, eigenvectors, skew-symmetric matrices, symmetric matrices, matrices with constant diagonal
CR Categories: 5.14
Language: Fortran

DESCRIPTION

The set of Fortran subroutines given here is an implementation of the algorithm [3] for finding the eigenvectors x and eigenvalues λ such that $Ax = \lambda x$, where A is a real skew-symmetric matrix or a real tridiagonal symmetric matrix with a constant diagonal. The algorithm uses only orthogonal similarity transformations and is believed to be the most efficient procedure available for computing all the eigenvalues or the complete eigensystem for the indicated classes of matrices.

The three subroutines of the algorithm and their functions are described as follows:

TRIZD. A subroutine that transforms an arbitrary real skew-symmetric matrix to skew-symmetric tridiagonal form using orthogonal similarity transformations, saving the pertinent information about these transformations.

IMZD. A subroutine that computes the eigenvalues and, optionally, the eigenvectors of a symmetric tridiagonal matrix with zeros on the diagonal or of a skew-symmetric tridiagonal matrix.

TBAKZD. A subroutine that computes the eigenvectors of an arbitrary real skew-symmetric matrix by back-transforming the eigenvectors of the corresponding skew-symmetric tridiagonal matrix determined by TRIZD. Subroutines TRIZD and TBAKZD are straightforward adaptations of Fortran subroutines TRED1 and TRBAK1 [2] (originally published as Algol procedures in [1]) which accomplish similar functions for arbitrary real symmetric matrices. A detailed description of subroutine IMZD is given by Ward and Gray [3].

To determine the complete eigensystem of a full skew-symmetric matrix, the user should issue a call to TRIZD, IMZD, and TBAKZD, in that order. If only the eigenvalues are desired, then calling TBAKZD is unnecessary. To determine the complete eigensystem or only the eigenvalues of a tridiagonal skew-symmetric matrix or a tridiagonal symmetric matrix with zero diagonals, the user should only issue a call to IMZD. (If $\{\lambda_i\}$ and $\{x_i\}$ are the eigenvalues and eigenvectors of the matrix A, a tridiagonal symmetric matrix with zero diagonals, then the eigenvalues and eigenvectors of $A + \alpha I$ are $\{\lambda_i + \alpha\}$ and $\{x_i\}$.)

Subroutines TRIZD, IMZD, and TBAKZD have been tested extensively on

Received 31 March 1977 and 26 October 1977.
Author's address: Computer Sciences Division, Mathematics and Statistics Research Department, Nuclear Division, Union Carbide Corporation, P.O. Box Y, Oak Ridge, TN 37830.

the IBM 360/91 computer at Oak Ridge National Laboratory using double precision arithmetic. Some of the test cases are presented in [3]. Since only orthogonal similarity transformations are used, the algorithm is numerically stable, that is, each computed eigenvalue and its corresponding eigenvector are exact for a matrix close to the original matrix.

REFERENCES

1. MARTIN, R.S., REINSCH, C., AND WILKINSON, J.H. Householder's tridiagonalization of a symmetric matrix. *Numer. Math. 11* (1968), 181–95.
2. SMITH, B.T., ET AL. Matrix eigensystem routines—EISPACK guide. In *Lecture Notes in Computer Science, Vol. 6,* Springer-Verlag, New York, 1974.
3. WARD, R.C., AND GRAY, L.J. Eigensystem computation for skew-symmetric matrices and a class of symmetric matrices. *ACM Trans. Math. Software 4,* 3 (Sept. 1978), 278–285.

ALGORITHM

```
C                                                                       MAIN0001
C ****                                                                  MAIN0002
C                                                                       MAIN0003
C  FUNCTION  -  THIS IS THE MAIN PROGRAM (DRIVER) FOR ILLUSTRATING THE  MAIN0004
C                 USE OF SUBROUTINES TRIZD, IMZD, AND TBAKZD.           MAIN0005
C                                                                       MAIN0006
C  REFERENCES - EIGENSYSTEM COMPUTATION FOR SKEW-SYMMETRIC MATRICES AND MAIN0007
C                 A CLASS OF SYMMETRIC MATRICES, WARD,R C AND GRAY,L J. MAIN0008
C                 TO APPEAR IN MANUSCRIPT SECTION OF ACM-TOMS.          MAIN0009
C                                                                       MAIN0010
C               AN ALGORITHM FOR COMPUTING THE EIGENSYSTEM OF SKEW-     MAIN0011
C                 SYMMETRIC MATRICES AND A CLASS OF SYMMETRIC MATRICES, MAIN0012
C                 TO APPEAR IN ALGORITHM SECTION OF ACM-TOMS.           MAIN0013
C                                                                       MAIN0014
C  REQUIRED FUNCTIONS FOR DRIVER AND SUBROUTINES - ABS,SIGN,SQRT,MOD    MAIN0015
C                                                                       MAIN0016
C ****                                                                  MAIN0017
C                                                                       MAIN0018
      REAL   A(6,6),Z(6,6),E(6)                                         MAIN0019
      REAL   CON                                                        MAIN0020
      INTEGER   I,J,N,IM1,JP1,MAX,IERR                                  MAIN0021
      LOGICAL MATZ,SKEW                                                 MAIN0022
      MAX = 6                                                           MAIN0023
      MATZ = .TRUE.                                                     MAIN0024
C                                                                       MAIN0025
C *** SET UP SKEW-SYMMETRIC TEST CASE                                   MAIN0026
C                                                                       MAIN0027
      WRITE (6,6000)                                                    MAIN0028
 6000 FORMAT (1H1,47HEIGENSYSTEM COMPUTATION OF SKEW-SYMMETRIC TEST ,   MAIN0029
     1        4HCASE//)                                                 MAIN0030
      N = 5                                                             MAIN0031
      SKEW = .TRUE.                                                     MAIN0032
C                                                                       MAIN0033
C *** READ AND PRINT TEST MATRIX                                        MAIN0034
C                                                                       MAIN0035
      WRITE (6,6001)                                                    MAIN0036
 6001 FORMAT (1H0,9X,11HTEST MATRIX/)                                   MAIN0037
      DO 10 I=1,N                                                       MAIN0038
         READ (5,6002) (A(I,J),J=1,N)                                   MAIN0039
         WRITE (6,6002) (A(I,J),J=1,N)                                  MAIN0040
 6002    FORMAT (6F6.0)                                                 MAIN0041
   10 CONTINUE                                                          MAIN0042
C                                                                       MAIN0043
C *** COMPUTE EIGENVALUES AND EIGENVECTORS                              MAIN0044
C                                                                       MAIN0045
      CALL TRIZD (MAX,N,A,E)                                            MAIN0046
      CALL IMZD (N,E,MATZ,SKEW,MAX,Z,IERR)                              MAIN0047
      IF (IERR .NE. 0) WRITE (6,6003) IERR                              MAIN0048
 6003 FORMAT (1H0/1H0,11HIMZD IERR =,I5)                                MAIN0049
      CALL TBAKZD (MAX,N,A,N,MAX,Z)                                     MAIN0050
C                                                                       MAIN0051
C *** PRINT EIGENVALUES AND EIGENVECTORS                                MAIN0052
C                                                                       MAIN0053
      WRITE (6,6004)                                                    MAIN0054
 6004 FORMAT (1H0/1H0,2X,11HEIGENVALUES,25X,12HEIGENVECTORS)            MAIN0055
      J = 0                                                             MAIN0056
   15    J = J + 1                                                      MAIN0057
```

```
      WRITE (6,6005)                                                    MAIN0058
 6005 FORMAT (/)                                                        MAIN0059
      IF (E(J) .EQ. 0.D0) GO TO 20                                      MAIN0060
      JP1 = J+1                                                         MAIN0061
      WRITE (6,6006) E(J),Z(1,J),Z(1,JP1)                               MAIN0062
 6006 FORMAT (1X,E15.8,4H * I,5X,E15.8,5H  +  ,E15.8,4H * I)            MAIN0063
      WRITE (6,6007) (Z(I,J),Z(I,JP1),I=2,N)                            MAIN0064
 6007 FORMAT (25X,E15.8,5H  +  ,E15.8,4H * I)                           MAIN0065
      WRITE (6,6005)                                                    MAIN0066
      CON = -Z(1,JP1)                                                   MAIN0067
      WRITE (6,6006) E(JP1),Z(1,J),CON                                  MAIN0068
      DO 18 I=2,N                                                       MAIN0069
         CON = -Z(I,JP1)                                                MAIN0070
         WRITE (6,6007) Z(I,J),CON                                      MAIN0071
   18 CONTINUE                                                          MAIN0072
      J = J + 1                                                         MAIN0073
      GO TO 30                                                          MAIN0074
   20 WRITE (6,6008) E(J),Z(1,J)                                        MAIN0075
 6008 FORMAT (1X,E15.8,9X,E15.8)                                        MAIN0076
      WRITE (6,6009) (Z(I,J),I=2,N)                                     MAIN0077
 6009 FORMAT (25X,E15.8)                                                MAIN0078
   30 IF (J .LT. N) GO TO 15                                            MAIN0079
C                                                                       MAIN0080
C *** SET UP TRIDIAGONAL, SYMMETRIC, ZERO DIAGONAL TEST CASE           MAIN0081
C                                                                       MAIN0082
      WRITE (6,6010)                                                    MAIN0083
 6010 FORMAT (1H1,50HEIGENSYSTEM COMPUTATION OF TRIDIAGONAL, SYMMETRIC,,MAIN0084
     1        24H ZERO DIAGONAL TEST CASE//)                           MAIN0085
      N = 6                                                             MAIN0086
      SKEW = .FALSE.                                                    MAIN0087
      DO 40 I=1,N                                                       MAIN0088
         E(I) = 1.                                                      MAIN0089
         DO 40 J=1,N                                                    MAIN0090
            A(I,J) = 0.                                                 MAIN0091
   40 CONTINUE                                                          MAIN0092
      DO 50 I=2,N                                                       MAIN0093
         IM1 = I-1                                                      MAIN0094
         A(I,IM1) = E(I)                                                MAIN0095
         A(IM1,I) = E(I)                                                MAIN0096
   50 CONTINUE                                                          MAIN0097
C                                                                       MAIN0098
C *** PRINT TEST MATRIX                                                 MAIN0099
C                                                                       MAIN0100
      WRITE (6,6011)                                                    MAIN0101
 6011 FORMAT (1H0,12X,11HTEST MATRIX/)                                  MAIN0102
      DO 60 I=1,N                                                       MAIN0103
         WRITE (6,6002) (A(I,J),J=1,N)                                  MAIN0104
   60 CONTINUE                                                          MAIN0105
C                                                                       MAIN0106
C *** COMPUTE EIGENVALUES AND EIGENVECTORS                              MAIN0107
C                                                                       MAIN0108
      CALL IMZD (N,E,MATZ,SKEW,MAX,Z,IERR)                              MAIN0109
      IF (IERR .NE. 0) WRITE (6,6003) IERR                              MAIN0110
C                                                                       MAIN0111
C *** PRINT EIGENVALUES AND EIGENVECTORS                                MAIN0112
C                                                                       MAIN0113
      WRITE (6,6012)                                                    MAIN0114
 6012 FORMAT (1H0/1H0,2X,11HEIGENVALUES,13X,12HEIGENVECTORS)            MAIN0115
      DO 70 J=1,N                                                       MAIN0116
         WRITE (6,6005)                                                 MAIN0117
         WRITE (6,6008) E(J),Z(1,J)                                     MAIN0118
         WRITE (6,6009) (Z(I,J),I=2,N)                                  MAIN0119
   70 CONTINUE                                                          MAIN0120
      STOP                                                              MAIN0121
      END                                                               MAIN0122

      SUBROUTINE TRIZD ( NA, N, A, E )                                  TRIZ0001
C                                                                       TRIZ0002
C ****                                                                  TRIZ0003
C                                                                       TRIZ0004
C  FUNCTION  -  REDUCES A REAL SKEW-SYMMETRIC MATRIX TO A SKEW-SYMMETRICTRIZ0005
C               TRIDIAGONAL MATRIX USING ORTHOGONAL SIMILARITY         TRIZ0006
C               TRANSFORMATIONS                                        TRIZ0007
C                                                                       TRIZ0008
C  PARAMETERS                                                           TRIZ0009
```

```
C                                                                       TRIZ0010
C     NA        - INPUT INTEGER SPECIFYING THE ROW DIMENSION OF A AS    TRIZ0011
C                   DECLARED IN THE CALLING PROGRAM DIMENSION STATEMENT TRIZ0012
C                                                                       TRIZ0013
C     N         - INPUT INTEGER SPECIFYING THE ORDER OF A               TRIZ0014
C                                                                       TRIZ0015
C     A(NA,N)   - ON INPUT, A CONTAINS THE REAL SKEW-SYMMETRIC MATRIX.  TRIZ0016
C                   ONLY THE STRICT LOWER TRIANGLE OF THE MATRIX NEED   TRIZ0017
C                   BE SUPPLIED.                                        TRIZ0018
C                 ON OUTPUT, A CONTAINS INFORMATION ABOUT THE ORTHOGONAL TRIZ0019
C                   TRANSFORMATIONS USED IN THE REDUCTION IN ITS FULL   TRIZ0020
C                   LOWER TRIANGLE. THE STRICT UPPER TRIANGLE OF A IS   TRIZ0021
C                   UNALTERED.                                          TRIZ0022
C                                                                       TRIZ0023
C     E(N)      - OUTPUT ARRAY CONTAINING THE LOWER SUBDIAGONAL ELEMENTS TRIZ0024
C                   OF THE TRIDIAGONAL MATRIX IN ITS LAST N-1 POSITIONS. TRIZ0025
C                   E(1) IS SET TO ZERO.                                TRIZ0026
C                                                                       TRIZ0027
C  REQUIRED FUNCTIONS - ABS,SIGN,SQRT                                   TRIZ0028
C                                                                       TRIZ0029
C ****                                                                  TRIZ0030
C                                                                       TRIZ0031
      REAL   A(NA,N),E(N)                                               TRIZ0032
      REAL   F,G,H,SCALE                                                TRIZ0033
      REAL   ABS,SIGN,SQRT                                              TRIZ0034
      INTEGER   I,J,K,L,II,JM1,JP1                                      TRIZ0035
C                                                                       TRIZ0036
      IF (N .EQ. 1) GO TO 230                                           TRIZ0037
C                                                                       TRIZ0038
C *** MAIN DO LOOP  I=N STEP -1 UNTIL 2                                 TRIZ0039
C                                                                       TRIZ0040
      DO 220 II = 2, N                                                  TRIZ0041
         I = N + 2 - II                                                 TRIZ0042
         L = I - 1                                                      TRIZ0043
         H = 0.                                                         TRIZ0044
         SCALE = 0.                                                     TRIZ0045
C                                                                       TRIZ0046
C *** NORMALIZE ROW                                                     TRIZ0047
C                                                                       TRIZ0048
         DO 100 K = 1, L                                                TRIZ0049
            SCALE = SCALE + ABS(A(I,K))                                 TRIZ0050
  100    CONTINUE                                                       TRIZ0051
C                                                                       TRIZ0052
         IF (SCALE .NE. 0.) GO TO 120                                   TRIZ0053
         E(I) = 0.                                                      TRIZ0054
         GO TO 215                                                      TRIZ0055
C                                                                       TRIZ0056
C *** COMPUTE ELEMENTS OF U VECTOR                                      TRIZ0057
C                                                                       TRIZ0058
  120    DO 130 K = 1, L                                                TRIZ0059
            A(I,K) = A(I,K) / SCALE                                     TRIZ0060
            H = H + A(I,K) * A(I,K)                                     TRIZ0061
  130    CONTINUE                                                       TRIZ0062
C                                                                       TRIZ0063
         F = A(I,L)                                                     TRIZ0064
         G = -SIGN(SQRT(H),F)                                           TRIZ0065
         E(I) = SCALE * G                                               TRIZ0066
         H = H - F * G                                                  TRIZ0067
         A(I,L) = F - G                                                 TRIZ0068
         IF (L .EQ. 1) GO TO 200                                        TRIZ0069
C                                                                       TRIZ0070
C *** COMPUTE ELEMENTS OF A*U/H                                         TRIZ0071
C                                                                       TRIZ0072
         DO 180 J = 1, L                                                TRIZ0073
            G = 0.                                                      TRIZ0074
            IF (J .EQ. 1) GO TO 150                                     TRIZ0075
            JM1 = J - 1                                                 TRIZ0076
C                                                                       TRIZ0077
            DO 140 K = 1, JM1                                           TRIZ0078
               G = G + A(J,K) * A(I,K)                                  TRIZ0079
  140       CONTINUE                                                    TRIZ0080
C                                                                       TRIZ0081
  150       IF (J .EQ. L) GO TO 170                                     TRIZ0082
            JP1 = J + 1                                                 TRIZ0083
C                                                                       TRIZ0084
            DO 160 K = JP1, L                                           TRIZ0085
```

```
                G = G - A(K,J) * A(I,K)                                 TRIZ0086
  160       CONTINUE                                                    TRIZ0087
C                                                                       TRIZ0088
  170       E(J) = G / H                                                TRIZ0089
  180    CONTINUE                                                       TRIZ0090
C                                                                       TRIZ0091
C *** COMPUTE REDUCED A                                                 TRIZ0092
C                                                                       TRIZ0093
         DO 190 J = 2, L                                                TRIZ0094
            F = A(I,J)                                                  TRIZ0095
            G = E(J)                                                    TRIZ0096
            JM1 = J - 1                                                 TRIZ0097
C                                                                       TRIZ0098
            DO 190 K = 1, JM1                                           TRIZ0099
               A(J,K) = A(J,K) + F * E(K) - G * A(I,K)                  TRIZ0100
  190    CONTINUE                                                       TRIZ0101
C                                                                       TRIZ0102
  200    DO 210 K = 1, L                                                TRIZ0103
            A(I,K) = SCALE * A(I,K)                                     TRIZ0104
  210    CONTINUE                                                       TRIZ0105
C                                                                       TRIZ0106
  215    A(I,I) = SCALE * SQRT(H)                                       TRIZ0107
  220 CONTINUE                                                          TRIZ0108
C                                                                       TRIZ0109
  230 E(1) = 0.                                                         TRIZ0110
      RETURN                                                            TRIZ0111
      END                                                               TRIZ0112

      SUBROUTINE IMZD ( N, E, MATZ, SKEW, NZ, Z, IERR )                 IMZD0001
C                                                                       IMZD0002
C ****                                                                  IMZD0003
C                                                                       IMZD0004
C  FUNCTION  -  COMPUTE THE EIGENVALUES AND OPTIONALLY THE EIGENVECTORS IMZD0005
C                 OF A SYMMETRIC TRIDIAGONAL MATRIX WITH ZERO DIAGONALS IMZD0006
C                 OR A SKEW-SYMMETRIC TRIDIAGONAL MATRIX USING AN       IMZD0007
C                 IMPLICIT QR-TYPE ITERATION                            IMZD0008
C                                                                       IMZD0009
C  PARAMETERS                                                           IMZD0010
C                                                                       IMZD0011
C     N         - INPUT INTEGER SPECIFYING THE ORDER OF THE TRIDIAGONAL  IMZD0012
C                   MATRIX                                              IMZD0013
C                                                                       IMZD0014
C     E(N)      - ON INPUT, ARRAY CONTAINING THE LOWER SUBDIAGONAL      IMZD0015
C                   ELEMENTS OF THE TRIDIAGONAL MATRIX IN ITS LAST N-1  IMZD0016
C                   POSITIONS. E(1) IS ARBITRARY.                       IMZD0017
C                 ON OUTPUT, ARRAY CONTAINS THE EIGENVALUES. THE NON-ZEROIMZD0018
C                   EIGENVALUES OCCUR IN PAIRS WITH OPPOSITE SIGNS AND  IMZD0019
C                   ARE FOUND IN ADJACENT LOCATIONS IN E. THE EIGENVALUESIMZD0020
C                   OF SYMMETRIC MATRICES ARE REAL AND THE EIGENVALUES  IMZD0021
C                   OF SKEW-SYMMETRIC MATRICES ARE PURELY IMAGINARY     IMZD0022
C                   COMPLEX NUMBERS. IF AN ERROR EXIT IS MADE, THE      IMZD0023
C                   EIGENVALUES ARE CORRECT FOR INDICES IERR+1,IERR+2...NIMZD0024
C                                                                       IMZD0025
C     MATZ      - INPUT LOGICAL VARIABLE SPECIFYING THE EIGENVECTOR     IMZD0026
C                   OPTION                                              IMZD0027
C                   = .TRUE.   EIGENVECTORS ARE TO BE COMPUTED          IMZD0028
C                   = .FALSE.  EIGENVECTORS ARE NOT TO BE COMPUTED      IMZD0029
C                                                                       IMZD0030
C     SKEW      - INPUT LOGICAL VARIABLE SPECIFYING TYPE OF INPUT MATRIX IMZD0031
C                   = .TRUE.   INPUT TRIDIAGONAL MATRIX IS SKEW-SYMMETRICIMZD0032
C                   = .FALSE.  INPUT TRIDIAGONAL MATRIX IS SYMMETRIC WITHIMZD0033
C                                ZERO DIAGONALS                         IMZD0034
C                   SKEW IS NOT REFERENCED IF MATZ = .FALSE.            IMZD0035
C                                                                       IMZD0036
C     NZ        - INPUT INTEGER SPECIFYING THE ROW DIMENSION OF Z AS    IMZD0037
C                   DECLARED IN THE CALLING PROGRAM DIMENSION STATEMENT IMZD0038
C                                                                       IMZD0039
C     Z(NZ,N)   - OUTPUT ARRAY CONTAINING THE ORTHOGONAL EIGENVECTORS   IMZD0040
C                   OF THE INPUT TRIDIAGONAL MATRIX. EIGENVECTORS CORRE- IMZD0041
C                   SPONDING TO ZERO EIGENVALUES ARE NORMALIZE TO UNIT   IMZD0042
C                   2-NORM (LENGTH) AND THOSE CORRESPONDING TO NON-ZERO  IMZD0043
C                   EIGENVALUES HAVE 2-NORM OF SQUARE ROOT 2. IF THE J-THIMZD0044
C                   EIGENVALUE IS ZERO OR REAL (I.E. E(J)), ITS EIGEN-   IMZD0045
C                   VECTOR IS FOUND IN THE J-TH COLUMN OF Z. IF THE J-TH IMZD0046
C                   EIGENVALUE IS IMAGINARY (I.E. E(J)*I) WITH E(J+1) =  IMZD0047
```

```
C                  -E(J), THE REAL PART OF ITS EIGENVECTOR IS FOUND IN  IMZD0048
C                  THE J-TH COLUMN OF Z AND ITS IMAGINARY PART FOUND IN IMZD0049
C                  THE (J+1)-TH COLUMN. IF AN ERROR EXIT IS MADE, Z     IMZD0050
C                  CONTAINS THE EIGENVECTORS ASSOCIATED WITH THE STORED IMZD0051
C                  EIGENVALUES.                                         IMZD0052
C                  Z IS NOT REFERENCED IF MATZ = .FALSE.                IMZD0053
C                                                                       IMZD0054
C     IERR     - OUTPUT ERROR CODE                                      IMZD0055
C                  = 0   NORMAL RETURN (ALL EIGENVALUES/VECTORS FOUND)  IMZD0056
C                  = J   IF THE J-TH EIGENVALUE HAS NOT BEEN DETERMINED IMZD0057
C                        AFTER 30 ITERATIONS                            IMZD0058
C                                                                       IMZD0059
C  REQUIRED FUNCTIONS - ABS,SIGN,SQRT,MOD                               IMZD0060
C                                                                       IMZD0061
C ****                                                                  IMZD0062
C                                                                       IMZD0063
      REAL   E(N),Z(NZ,N)                                               IMZD0064
      REAL   F,G,Q,C,S,R,P,TEST,TMAG                                    IMZD0065
      REAL   ABS,SIGN,SQRT                                              IMZD0066
      INTEGER   I,J,K,L,M,LO,LOM1,MO,LS,IM1,JP1,KM1,KP1,LM1,MM1,        IMZD0067
     1          IP3,IERR,IEO,ITS,IP1                                    IMZD0068
      INTEGER   MOD                                                     IMZD0069
      LOGICAL MATZ,SKEW                                                 IMZD0070
C                                                                       IMZD0071
      IF (.NOT. MATZ) GO TO 115                                         IMZD0072
C                                                                       IMZD0073
C *** PLACE IDENTITY MATRIX IN Z                                        IMZD0074
C                                                                       IMZD0075
      DO 110 I = 1, N                                                   IMZD0076
         DO 100 J = 1, N                                                IMZD0077
            Z(I,J) = 0.                                                 IMZD0078
  100    CONTINUE                                                       IMZD0079
         Z(I,I) = 1.                                                    IMZD0080
  110 CONTINUE                                                          IMZD0081
C                                                                       IMZD0082
  115 IERR = 0                                                          IMZD0083
      M = N                                                             IMZD0084
      MM1 = M - 1                                                       IMZD0085
      E(1) = 0.                                                         IMZD0086
      ITS = 0                                                           IMZD0087
C                                                                       IMZD0088
  120 IF (M .LT. 2) GO TO 370                                           IMZD0089
      MO = M                                                            IMZD0090
C                                                                       IMZD0091
C *** SEARCH FOR NEXT SUBMATRIX TO SOLVE  (MATRIX SPLITTING)            IMZD0092
C                                                                       IMZD0093
      F = 0.                                                            IMZD0094
      DO 130 I = 1, MM1                                                 IMZD0095
         J = M - I                                                      IMZD0096
         JP1 = J + 1                                                    IMZD0097
         G = ABS(E(JP1))                                                IMZD0098
         TMAG = ABS(E(J)) + F                                           IMZD0099
         TEST = TMAG + G                                                IMZD0100
         IF (TEST .EQ. TMAG) GO TO 140                                  IMZD0101
         F = G                                                          IMZD0102
  130 CONTINUE                                                          IMZD0103
      JP1 = 1                                                           IMZD0104
C                                                                       IMZD0105
  140 LO = JP1 + 1                                                      IMZD0106
      LOM1 = JP1                                                        IMZD0107
      IF (LOM1 .EQ. M) GO TO 290                                        IMZD0108
      IF (.NOT. MATZ) GO TO 160                                         IMZD0109
      IF (.NOT. SKEW) GO TO 160                                         IMZD0110
C                                                                       IMZD0111
C *** PLACE CORRECT SIGN ON IDENTITY DIAGONALS                          IMZD0112
C                                                                       IMZD0113
      DO 150 I = LOM1, M, 4                                             IMZD0114
         Z(I,I) = -Z(I,I)                                               IMZD0115
         IP3 = I + 3                                                    IMZD0116
         IF (IP3 .GT. M) GO TO 160                                      IMZD0117
         Z(IP3,IP3) = -Z(IP3,IP3)                                       IMZD0118
  150 CONTINUE                                                          IMZD0119
C                                                                       IMZD0120
  160 IF (LO .EQ. M) GO TO 300                                          IMZD0121
      IEO = M - LO                                                      IMZD0122
      IEO = MOD(IEO,2)                                                  IMZD0123
```

```
      L = L0                                                          IMZD0124
      IF (IEO .EQ. 0) GO TO 230                                       IMZD0125
C                                                                     IMZD0126
C *** FIND ZERO EIGENVALUE OF ODD ORDERED SUBMATRICES                 IMZD0127
C                                                                     IMZD0128
      C = 0.                                                          IMZD0129
      S = -1.                                                         IMZD0130
      DO 190 I = L0, MM1, 2                                           IMZD0131
         K = MM1 + L0 - I                                             IMZD0132
         KP1 = K + 1                                                  IMZD0133
         Q = -S * E(KP1)                                              IMZD0134
         E(KP1) = C * E(KP1)                                          IMZD0135
         IF (ABS(E(K)) .GT. ABS(Q)) GO TO 170                         IMZD0136
         C = E(K) / Q                                                 IMZD0137
         R = SQRT(C*C + 1.)                                           IMZD0138
         E(K) = Q * R                                                 IMZD0139
         S = 1. / R                                                   IMZD0140
         C = C * S                                                    IMZD0141
         GO TO 180                                                    IMZD0142
  170    S = Q / E(K)                                                 IMZD0143
         R = SQRT(1. + S*S)                                           IMZD0144
         E(K) = E(K) * R                                              IMZD0145
         C = 1. / R                                                   IMZD0146
         S = S * C                                                    IMZD0147
  180    IF (.NOT. MATZ) GO TO 190                                    IMZD0148
C                                                                     IMZD0149
C *** ACCUMULATE TRANSFORMATIONS FOR EIGENVECTORS                     IMZD0150
C                                                                     IMZD0151
         KM1 = K - 1                                                  IMZD0152
         Z(KM1,M) = -S * Z(KM1,KM1)                                   IMZD0153
         Z(KM1,KM1) = C * Z(KM1,KM1)                                  IMZD0154
         DO 185 J = KP1, M, 2                                         IMZD0155
            Z(J,KM1) = S * Z(J,M)                                     IMZD0156
            Z(J,M) = C * Z(J,M)                                       IMZD0157
  185    CONTINUE                                                     IMZD0158
C                                                                     IMZD0159
  190 CONTINUE                                                        IMZD0160
      M = MM1                                                         IMZD0161
      MM1 = M - 1                                                     IMZD0162
      IF (L0 .EQ. M) GO TO 300                                        IMZD0163
C                                                                     IMZD0164
C *** CHECK FOR CONVERGENCE OR SMALL SUBDIAGONAL ELEMENT              IMZD0165
C                                                                     IMZD0166
  200 DO 210 I = L0, MM1, 2                                           IMZD0167
         K = MM1 + L0 - I                                             IMZD0168
         L = K + 1                                                    IMZD0169
         TMAG = ABS(E(L)) + ABS(E(K-1))                               IMZD0170
         TEST = TMAG + E(K)                                           IMZD0171
         IF (TEST .EQ. TMAG) GO TO 220                                IMZD0172
  210 CONTINUE                                                        IMZD0173
      L = L0                                                          IMZD0174
  220 IF (L .EQ. M) GO TO 300                                         IMZD0175
C                                                                     IMZD0176
C *** FORM SHIFT                                                      IMZD0177
C                                                                     IMZD0178
  230 ITS = ITS + 1                                                   IMZD0179
      IF (ITS .GT. 30) GO TO 360                                      IMZD0180
      F = E(M-3)                                                      IMZD0181
      G = E(M-2)                                                      IMZD0182
      C = E(MM1)                                                      IMZD0183
      S = E(M)                                                        IMZD0184
      P = ((C-F) * (C+F) + (S-G) * (S+G)) / (2. * G * C)              IMZD0185
      R = SQRT(P*P + 1.)                                              IMZD0186
      Q = (G / (P + SIGN(R,P))) - C                                   IMZD0187
      F = E(L)                                                        IMZD0188
      LM1 = L - 1                                                     IMZD0189
      E(LM1) = ((F-S) * (F+S) + C * Q) / F                            IMZD0190
C                                                                     IMZD0191
C *** PERFORM ONE IMPLICIT QR ITERATION ON CHOLESKY FACTOR            IMZD0192
C                                                                     IMZD0193
      LS = LOM1                                                       IMZD0194
      C = 1.                                                          IMZD0195
      S = 1.                                                          IMZD0196
      DO 280 I = L, MM1                                               IMZD0197
         IP1 = I + 1                                                  IMZD0198
         IM1 = I - 1                                                  IMZD0199
```

```
            Q = S * E(IP1)                                                  IMZD0200
            E(IP1) = C * E(IP1)                                             IMZD0201
            IF (ABS(E(IM1)) .GT. ABS(Q)) GO TO 240                          IMZD0202
            C = E(IM1) / Q                                                  IMZD0203
            R = SQRT(C*C + 1.)                                              IMZD0204
            E(IM1) = Q * R                                                  IMZD0205
            S = 1. / R                                                      IMZD0206
            C = C * S                                                       IMZD0207
            GO TO 250                                                       IMZD0208
  240       S = Q / E(IM1)                                                  IMZD0209
            R = SQRT(1. + S*S)                                              IMZD0210
            E(IM1) = E(IM1) * R                                             IMZD0211
            C = 1. / R                                                      IMZD0212
            S = S * C                                                       IMZD0213
  250       F = E(IP1)                                                      IMZD0214
            E(IP1) = -S * E(I) + C * F                                      IMZD0215
            E(I) = C * E(I) + S * F                                         IMZD0216
            IF (.NOT. MATZ) GO TO 280                                       IMZD0217
C                                                                           IMZD0218
C *** ACCUMULATE TRANSFORMATIONS FOR EIGENVECTORS                           IMZD0219
C                                                                           IMZD0220
            DO 260 J = LS, M0, 2                                            IMZD0221
               F = Z(J,IP1)                                                 IMZD0222
               Z(J,IP1) = -S * Z(J,IM1) + C * F                             IMZD0223
               Z(J,IM1) = C * Z(J,IM1) + S * F                              IMZD0224
  260       CONTINUE                                                        IMZD0225
            IF (LS .EQ. LOM1) GO TO 270                                     IMZD0226
            LS = LOM1                                                       IMZD0227
            GO TO 280                                                       IMZD0228
  270       LS = L0                                                         IMZD0229
  280 CONTINUE                                                              IMZD0230
            E(LM1) = 0.                                                     IMZD0231
            GO TO 200                                                       IMZD0232
C                                                                           IMZD0233
C *** ITERATION CONVERGED TO ONE ZERO EIGENVALUE                            IMZD0234
C                                                                           IMZD0235
  290 E(M) = 0.                                                             IMZD0236
            M = MM1                                                         IMZD0237
            GO TO 310                                                       IMZD0238
C                                                                           IMZD0239
C *** ITERATION CONVERGED TO EIGENVALUE PAIR                                IMZD0240
C                                                                           IMZD0241
  300 E(MM1) = E(M)                                                         IMZD0242
            E(M) = -E(M)                                                    IMZD0243
            M = M - 2                                                       IMZD0244
C                                                                           IMZD0245
  310 ITS = 0                                                               IMZD0246
            MM1 = M - 1                                                     IMZD0247
            IF (M .GT. L0) GO TO 200                                        IMZD0248
            IF (M .EQ. L0) GO TO 300                                        IMZD0249
            IF (.NOT. MATZ) GO TO 120                                       IMZD0250
            IF (SKEW) GO TO 120                                             IMZD0251
C                                                                           IMZD0252
C *** COMPUTE EIGENVECTORS FROM ORTHONORMAL COLUMNS OF Z IF NOT SKEW         IMZD0253
C                                                                           IMZD0254
  320 K = M0                                                                IMZD0255
  330 IF (E(K) .EQ. 0.) GO TO 350                                           IMZD0256
            KM1 = K - 1                                                     IMZD0257
            DO 340 J = LOM1, M0, 2                                          IMZD0258
               Z(J,K) = Z(J,KM1)                                            IMZD0259
               F = Z(J+1,K)                                                 IMZD0260
               Z(J+1,KM1) = F                                               IMZD0261
               Z(J+1,K) = -F                                                IMZD0262
  340 CONTINUE                                                              IMZD0263
            K = KM1                                                         IMZD0264
  350 K = K-1                                                               IMZD0265
            IF (K .GT. LOM1) GO TO 330                                      IMZD0266
            IF (IERR .NE. 0) GO TO 370                                      IMZD0267
            GO TO 120                                                       IMZD0268
C                                                                           IMZD0269
C *** ERROR EXIT                                                            IMZD0270
C                                                                           IMZD0271
  360 IERR = M                                                              IMZD0272
            IF (.NOT. MATZ) GO TO 370                                       IMZD0273
            IF (.NOT. SKEW) GO TO 320                                       IMZD0274
C                                                                           IMZD0275
  370 RETURN                                                                IMZD0276
            END                                                             IMZD0277
```

```
      SUBROUTINE TBAKZD ( NA, N, A, M, NZ, Z )                          TBAK0001
C                                                                       TBAK0002
C ****                                                                  TBAK0003
C                                                                       TBAK0004
C  FUNCTION  -  FORMS THE EIGENVECTORS OF A REAL SKEW-SYMMETRIC MATRIX  TBAK0005
C                 BY BACK TRANSFORMING THOSE OF THE CORRESPONDING SKEW- TBAK0006
C                 SYMMETRIC TRIDIAGONAL MATRIX DETERMINED BY  TRIZD    TBAK0007
C                                                                       TBAK0008
C  PARAMETERS                                                           TBAK0009
C                                                                       TBAK0010
C     NA       - INPUT INTEGER SPECIFYING THE ROW DIMENSION OF A        TBAK0011
C                  AS DECLARED IN CALLING PROGRAM DIMENSION STATEMENT   TBAK0012
C                                                                       TBAK0013
C     N        - INPUT INTEGER SPECIFYING THE ORDER OF A                TBAK0014
C                                                                       TBAK0015
C     A(NA,N)  - INPUT ARRAY CONTAINING INFORMATION ABOUT THE ORTHOGONALTBAK0016
C                  TRANSFORMATIONS USED IN THE REDUCTION BY  TRIZD  IN  TBAK0017
C                  ITS FULL LOWER TRIANGLE                              TBAK0018
C                                                                       TBAK0019
C     M        - INPUT INTEGER SPECIFYING THE NUMBER OF EIGENVECTORS TO TBAK0020
C                  BE BACK TRANSFORMED                                  TBAK0021
C                                                                       TBAK0022
C     NZ       - INPUT INTEGER SPECIFYING THE ROW DIMENSION OF Z AS     TBAK0023
C                  DECLARED IN CALLING PROGRAM DIMENSION STATEMENT      TBAK0024
C                                                                       TBAK0025
C     Z(NZ,M)  - ON INPUT, Z CONTAINS THE REAL AND IMAGINARY (IF        TBAK0026
C                  COMPLEX) PARTS OF THE EIGENVECTORS TO BE BACK TRANS- TBAK0027
C                  FORMED IN ITS FIRST M COLUMNS                        TBAK0028
C                ON OUTPUT, Z CONTAINS THE REAL AND IMAGINARY (IF       TBAK0029
C                  COMPLEX) PARTS OF THE TRANSFORMED EIGENVECTORS IN    TBAK0030
C                  ITS FIRST M COLUMNS                                  TBAK0031
C                                                                       TBAK0032
C ****                                                                  TBAK0033
C                                                                       TBAK0034
      REAL   A(NA,N),Z(NZ,M)                                            TBAK0035
      REAL   H,S                                                        TBAK0036
      INTEGER   I,J,K,L                                                 TBAK0037
C                                                                       TBAK0038
      IF (M .EQ. 0) GO TO 140                                           TBAK0039
      IF (N .EQ. 1) GO TO 140                                           TBAK0040
C                                                                       TBAK0041
      DO 130 I = 2, N                                                   TBAK0042
         L = I - 1                                                      TBAK0043
         H = A(I,I)                                                     TBAK0044
         IF (H .EQ. 0.) GO TO 130                                       TBAK0045
C                                                                       TBAK0046
         DO 120 J = 1, M                                                TBAK0047
            S = 0.                                                      TBAK0048
C                                                                       TBAK0049
            DO 100 K = 1, L                                             TBAK0050
               S = S + A(I,K) * Z(K,J)                                  TBAK0051
  100       CONTINUE                                                    TBAK0052
C                                                                       TBAK0053
            S = (S / H) / H                                             TBAK0054
C                                                                       TBAK0055
            DO 110 K = 1, L                                             TBAK0056
               Z(K,J) = Z(K,J) - S * A(I,K)                             TBAK0057
  110       CONTINUE                                                    TBAK0058
C                                                                       TBAK0059
  120    CONTINUE                                                       TBAK0060
C                                                                       TBAK0061
  130 CONTINUE                                                          TBAK0062
C                                                                       TBAK0063
  140 RETURN                                                            TBAK0064
      END                                                               TBAK0065
   0.    2.   -2.    0.   -4.
  -2.    0.    4.   -5.    3.
   2.   -4.    0.    1.    1.
   0.    5.   -1.    0.    2.
   4.   -3.   -1.   -2.    0.
```

ALGORITHM 531
Contour Plotting [J6]

WILLIAM V. SNYDER
Jet Propulsion Laboratory

Key Words and Phrases: contour plotting
CR Categories: 8.2
Language: Fortran

DESCRIPTION

Given a two-dimensional array of samples of a surface, contour values, and a subroutine to draw lines, the subroutine GCONTR determines sequences of points in the plane which may be used to draw contours through equal values of the surface.

A contour plotting algorithm may be constructed by following contours from some starting point until they either close or intersect a boundary, or by examining each cell of the grid in turn and drawing all contours found inside the cell. The advantages of contour following are that contour labeling is relatively easy and that the pen of an incremental plotter does not move about as much without writing anything. The advantages of methods which draw all contours found inside a cell are that less auxiliary storage is needed and that each cell can be completely processed before going on to the next, thereby allowing generation of contours over a much larger array than can be accommodated in main memory at one time.

GCONTR is of the type which follows contours. On a representative problem, a program using GCONTR generated about 11,000 plotter commands with about 57,000 commands generated by a program using a cellular method. For this problem, grid lines, user identification, and a table of contour values required about 36,000 additional commands.

Since the data used by contour plotting programs are presented at discrete points and since the contour values are not necessarily equal to the dependent variable values at the nodal points of the mesh, some method must be used to estimate the point at which a contour value intersects the edge of a cell. If one uses nonlinear interpolation, there is the possibility of multiple intersections of a contour with an edge and the possibility of closed contours which intersect no edges; if one wants to follow the interpolated surface faithfully, drawing the contour entails finding zeros of a nonlinear bivariate function. Linear interpolation

Received 17 June 1977 and 8 November 1977.

This work was sponsored by the National Aeronautics and Space Administration under Contract NAS 7-100; this paper presents the results of one phase of research carried out at the Jet Propulsion Laboratory. The work was carried out using a Univac 1108 computer, CalComp model 565 drum plotters, a Calcomp model 1675 Com, and Tektronix 4000 series interactive graphics terminals.
Author's address: Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91103.

has the drawback that extrema occur only at nodal points. GCONTR uses linear interpolation. If this does not produce sufficiently smooth, accurate contours, we recommend computing more function values or interpolatory subtabulation to a finer mesh. If the required storage is larger than desired, the independent variable plane is easily divided into separately processed segments. GCONTR does not provide automatic subtabulation because different methods work better for different problems. Since GCONTR uses a contour following method, any implementation of automatic subtabulation would necessarily have a higher cost than a reasonable user implementation.

Once the points of intersection of a contour with edges have been determined, one must still decide how to draw the contour line between such intersections. We recommend that straight lines be used. Most other curves can cause balloons and crossing of contours of different values.

We now discuss the features which distinguish GCONTR from the methods of Cottafava and Le Moli [1] and Crane [2]. GCONTR and the algorithms described in [1] and [2] are similar in that all are contour following methods, the data are assumed to be presented on the nodes of a topologically rectangular grid, and linear interpolation is used along the edges of a cell.

GCONTR and algorithm [2] begin following a contour as soon as an edge of a cell of the coordinate grid is found which the contour line intersects. Edges intersected by the contour are flagged in an integer array in [2] and by marks in a bitmap in GCONTR. In algorithm [1] a preliminary detection of all intersections of edges and contours is done and then the contours are drawn. The details of the bookkeeping are not described in [1].

Algorithms [1] and [2] start the contour at the first edge found while searching along rows or columns of the data array. GCONTR reduces pen-up travel by searching along a rectangular spiral starting at the current pen position. All three algorithms draw all contours which intersect boundaries before drawing any contours which do not.

Algorithm [1] treats the case of a contour intersecting all four edges of a cell in a way which depends on the fixed order of examining the edges. If a cell is approached from different directions, different contours will be drawn. Algorithm [2] forces the contours to cross inside the cell. When several different contours intersect all four edges, the resulting pattern looks like an asterisk. GCONTR examines the linear interpolate along the "top" and "bottom" edges of the cell. If the interpolate on the top edge is less than the interpolate on the bottom edge, the contour line is drawn top-to-left and bottom-to-right. Otherwise, the contour line is drawn top-to-right and bottom-to-left. This methods selects the same contours if the axes are exchanged. To show this, note that the use of linear interpolation along the edges of the unit cell implies that there is some estimated function value F_x which occurs at the same abscissa on the top and bottom edges, and some estimated function value F_y which occurs at the same ordinate on the "left" and "right" edges. It is easy to show that $F_x = F_y$. Therefore, the contour of value F_x is the only contour which crosses inside the cell, and it divides the cell into four rectangular regions. In two of these regions the esimated function value is less than F_x. In the other two, the estimated function value is greater than F_x.

Algorithms [1] and [2] consider all data in the array. GCONTR provides a means whereby the user may designate excluded elements of the array. The edges connecting excluded elements to elements not excluded are not examined. This feature has been found to be very useful in practice, as data are frequently not presented on a complete rectangular grid. When using methods which do not have this feature, one frequently finds extraneous contours drawn between the regions containing data and the regions containing no data.

The output method is not described in [1]. The output from algorithm [2] is an ordered set of points in an array which define the contour lines to be drawn. GCONTR calls a line drawing subroutine provided by the user. If the user wishes to take the risks noted above associated with using other than straight lines for

drawing the contours, points may be stored in the line drawing subroutine, and a method such as is described in [3] may be used to generate smooth contours. Transformations may be applied to the output of algorithm [2] and to the points provided by GCONTR to the line drawing subroutine for such purposes as coordinate system conversion. Note that neither algorithm [2] nor GCONTR require the grid lines to be straight or uniformly spaced.

REFERENCES

1. COTTAFAVA, G., AND LE MOLI, G. Automatic contour map. *Comm. ACM 12*, 7 (July 1969), 386-391.
2. CRANE, C.M. Contour plotting for functions specified at nodal points of an irregular mesh based on an arbitrary two parameter co-ordinate system. *Comptr. J. 15* (1972), 382-384.
3. MCCONALOGUE, D.J. An automatic french-curve procedure for use with an incremental plotter. *Comptr. J. 14* (1971), 207-209.

ALGORITHM

```
      SUBROUTINE FILL0(BITMAP, N)                                        FIL   10
C                                                                        FIL   20
C     FILL THE FIRST N BITS OF BITMAP WITH ZEROES.                       FIL   30
C                                                                        FIL   40
      INTEGER BITMAP(1), N                                               FIL   50
C                                                                        FIL   60
      DATA NBPW /35/                                                     FIL   70
C     NBPW IS THE MINIMUM NUMBER OF SIGNIFICANT BITS PER WORD USED       FIL   80
C     BY INTEGER ARITHMETIC.  THIS IS USUALLY ONE LESS THAN THE          FIL   90
C     ACTUAL NUMBER OF BITS PER WORD, BUT AN IMPORTANT EXCEPTION IS      FIL  100
C     THE CDC-6000 SERIES OF MACHINES, WHERE NBPW SHOULD BE 48.          FIL  110
C                                                                        FIL  120
      LOOP = N/NBPW                                                      FIL  130
      NBLW = MOD(N,NBPW)                                                 FIL  140
      IF (LOOP.EQ.0) GO TO 20                                            FIL  150
      DO 10 I=1,LOOP                                                     FIL  160
        BITMAP(I) = 0                                                    FIL  170
   10 CONTINUE                                                           FIL  180
   20 IF (NBLW.NE.0) BITMAP(LOOP+1) = MOD(BITMAP(LOOP+1),2**(NBPW-NBLW)  FIL  190
     * )                                                                 FIL  200
      RETURN                                                             FIL  210
      END                                                                FIL  220

      SUBROUTINE MARK1(BITMAP, N)                                        MAR   10
C                                                                        MAR   20
C     PUT A ONE IN THE NTH BIT OF BITMAP.                                MAR   30
C                                                                        MAR   40
      INTEGER BITMAP(1), N                                               MAR   50
C                                                                        MAR   60
      DATA NBPW /35/                                                     MAR   70
C     NBPW IS THE MINIMUM NUMBER OF SIGNIFICANT BITS PER WORD USED       MAR   80
C     BY INTEGER ARITHMETIC.  THIS IS USUALLY ONE LESS THAN THE          MAR   90
C     ACTUAL NUMBER OF BITS PER WORD, BUT AN IMPORTANT EXCEPTION IS      MAR  100
C     THE CDC-6000 SERIES OF MACHINES, WHERE NBPW SHOULD BE 48.          MAR  110
C                                                                        MAR  120
      NWORD = (N-1)/NBPW                                                 MAR  130
      NBIT = MOD(N-1,NBPW)                                               MAR  140
      I = 2**(NBPW-NBIT-1)                                               MAR  150
      BITMAP(NWORD+1) = BITMAP(NWORD+1) + I*(1-MOD(BITMAP(NWORD+1)/I,2)  MAR  160
     * )                                                                 MAR  170
      RETURN                                                             MAR  180
      END                                                                MAR  190

      FUNCTION IGET(BITMAP, N)                                           IGE   10
C                                                                        IGE   20
C     IGET=0 IF THE NTH BIT OF BITMAP IS ZERO, ELSE IGET IS ONE.         IGE   30
C                                                                        IGE   40
      INTEGER BITMAP(1), N                                               IGE   50
C                                                                        IGE   60
      DATA NBPW /35/                                                     IGE   70
C     NBPW IS THE MINIMUM NUMBER OF SIGNIFICANT BITS PER WORD USED       IGE   80
C     BY INTEGER ARITHMETIC.  THIS IS USUALLY ONE LESS THAN THE          IGE   90
C     ACTUAL NUMBER OF BITS PER WORD, BUT AN IMPORTANT EXCEPTION IS      IGE  100
C     THE CDC-6000 SERIES OF MACHINES, WHERE NBPW SHOULD BE 48.          IGE  110
C                                                                        IGE  120
```

```
      NWORD = (N-1)/NBPW                                                IGE  130
      NBIT = MOD(N-1,NBPW)                                              IGE  140
      IGET = MOD(BITMAP(NWORD+1)/2**(NBPW-NBIT-1),2)                    IGE  150
      RETURN                                                            IGE  160
      END                                                               IGE  170


      SUBROUTINE GCONTR(Z, NRZ, NX, NY, CV, NCV, ZMAX, BITMAP, DRAW)    GCO   10
C                                                                       GCO   20
C     THIS SUBROUTINE DRAWS A CONTOUR THROUGH EQUAL VALUES OF AN ARRAY. GCO   30
C                                                                       GCO   40
C     *****     FORMAL ARGUMENTS     **********************************GCO   50
C                                                                       GCO   60
C     Z IS THE ARRAY FOR WHICH CONTOURS ARE TO BE DRAWN.  THE ELEMENTS  GCO   70
C     OF Z ARE ASSUMED TO LIE UPON THE NODES OF A TOPOLOGICALLY         GCO   80
C     RECTANGULAR COORDINATE SYSTEM - E.G. CARTESIAN, POLAR (EXCEPT     GCO   90
C     THE ORIGIN), ETC.                                                 GCO  100
C                                                                       GCO  110
C     NRZ IS THE NUMBER OF ROWS DECLARED FOR Z IN THE CALLING PROGRAM.  GCO  120
C                                                                       GCO  130
C     NX IS THE LIMIT FOR THE FIRST SUBSCRIPT OF Z.                     GCO  140
C                                                                       GCO  150
C     NY IS THE LIMIT FOR THE SECOND SUBSCRIPT OF Z.                    GCO  160
C                                                                       GCO  170
C     CV ARE THE VALUES OF THE CONTOURS TO BE DRAWN.                    GCO  180
C                                                                       GCO  190
C     NCV IS THE NUMBER OF CONTOUR VALUES IN CV.                        GCO  200
C                                                                       GCO  210
C     ZMAX IS THE MAXIMUM VALUE OF Z FOR CONSIDERATION.  A VALUE OF     GCO  220
C     Z(I,J) GREATER THAN ZMAX IS A SIGNAL THAT THAT POINT AND THE      GCO  230
C     GRID LINE SEGMENTS RADIATING FROM THAT POINT TO IT'S NEIGHBORS    GCO  240
C     ARE TO BE EXCLUDED FROM CONTOURING.                               GCO  250
C                                                                       GCO  260
C     BITMAP IS A WORK AREA LARGE ENOUGH TO HOLD 2*NX*NY*NCV BITS.  IT  GCO  270
C     IS ACCESSED BY LOW-LEVEL ROUTINES, WHICH ARE DESCRIBED BELOW.     GCO  280
C     LET J BE THE NUMBER OF USEFUL BITS IN EACH WORD OF BITMAP,        GCO  290
C     AS DETERMINED BY THE USER MACHINE AND IMPLEMENTATION OF           GCO  300
C     THE BITMAP MANIPULATION SUBPROGRAMS DESCRIBED BELOW.  THEN        GCO  310
C     THE NUMBER OF WORDS REQUIRED FOR THE BITMAP IS THE FLOOR OF       GCO  320
C         (2*NX*NY*NCV+J-1)/J.                                          GCO  330
C                                                                       GCO  340
C     DRAW IS A USER-PROVIDED SUBROUTINE USED TO DRAW CONTOURS.         GCO  350
C     THE CALLING SEQUENCE FOR DRAW IS:                                 GCO  360
C                                                                       GCO  370
C         CALL DRAW (X,Y,IFLAG)                                         GCO  380
C         LET NX = INTEGER PART OF X, FX = FRACTIONAL PART OF X.        GCO  390
C         THEN X SHOULD BE INTERPRETED SUCH THAT INCREASES IN NX        GCO  400
C         CORRESPOND TO INCREASES IN THE FIRST SUBSCRIPT OF Z, AND      GCO  410
C         FX IS THE FRACTIONAL DISTANCE FROM THE ABSCISSA CORRESPONDING GCO  420
C         TO NX TO THE ABSCISSA CORRESPONDING TO NX+1,                  GCO  430
C         AND Y SHOULD BE INTERPRETED SIMILARLY FOR THE SECOND          GCO  440
C         SUBSCRIPT OF Z.                                               GCO  450
C         THE LOW-ORDER DIGIT OF IFLAG WILL HAVE ONE OF THE VALUES:     GCO  460
C             1 - CONTINUE A CONTOUR,                                   GCO  470
C             2 - START A CONTOUR AT A BOUNDARY,                        GCO  480
C             3 - START A CONTOUR NOT AT A BOUNDARY,                    GCO  490
C             4 - FINISH A CONTOUR AT A BOUNDARY,                       GCO  500
C             5 - FINISH A CLOSED CONTOUR (NOT AT A BOUNDARY).          GCO  510
C                 NOTE THAT REQUESTS 1, 4 AND 5 ARE FOR PEN-DOWN        GCO  520
C                 MOVES, AND THAT REQUESTS 2 AND 3 ARE FOR PEN-UP       GCO  530
C                 MOVES.                                                GCO  540
C             6 - SET X AND Y TO THE APPROXIMATE 'PEN' POSITION, USING  GCO  550
C                 THE NOTATION DISCUSSED ABOVE.  THIS CALL MAY BE       GCO  560
C                 IGNORED, THE RESULT BEING THAT THE 'PEN' POSITION     GCO  570
C                 IS TAKEN TO CORRESPOND TO Z(1,1).                     GCO  580
C         IFLAG/10 IS THE CONTOUR NUMBER.                               GCO  590
C                                                                       GCO  600
C     *****     EXTERNAL SUBPROGRAMS     ******************************GCO  610
C                                                                       GCO  620
C     DRAW IS THE USER-SUPPLIED LINE DRAWING SUBPROGRAM DESCRIBED ABOVE.GCO  630
C     DRAW MAY BE SENSITIVE TO THE HOST COMPUTER AND TO THE PLOT DEVICE.GCO  640
C     FILLO IS USED TO FILL A BITMAP WITH ZEROES.  CALL FILLO (BITMAP,N)GCO  650
C     FILLS THE FIRST N BITS OF BITMAP WITH ZEROES.                     GCO  660
C     MARK1 IS USED TO PLACE A 1 IN A SPECIFIC BIT OF THE BITMAP.       GCO  670
C     CALL MARK1 (BITMAP,N) PUTS A 1 IN THE NTH BIT OF THE BITMAP.      GCO  680
```

```
C     IGET IS USED TO DETERMINE THE SETTING OF A PARTICULAR BIT IN THE  GCO  690
C     BITMAP.  I=IGET(BITMAP,N) SETS I TO ZERO IF THE NTH BIT OF THE    GCO  700
C     BITMAP IS ZERO, AND SETS I TO ONE IF THE NTH BIT IS ONE.          GCO  710
C     FILL0, MARK1 AND IGET ARE MACHINE SENSITIVE.                      GCO  720
C                                                                       GCO  730
C     ************************************************************************GCO  740
C                                                                       GCO  750
      REAL Z(NRZ,1), CV(1)                                            GCO  760
      INTEGER BITMAP(1)                                               GCO  770
      INTEGER L1(4), L2(4), IJ(2)                                     GCO  780
C                                                                       GCO  790
C     L1 AND L2 CONTAIN LIMITS USED DURING THE SPIRAL SEARCH FOR THE    GCO  800
C     BEGINNING OF A CONTOUR.                                           GCO  810
C     IJ STORES SUBCRIPTS USED DURING THE SPIRAL SEARCH.                GCO  820
C                                                                       GCO  830
      INTEGER I1(2), I2(2), I3(6)                                     GCO  840
C                                                                       GCO  850
C     I1, I2 AND I3 ARE USED FOR SUBSCRIPT COMPUTATIONS DURING THE      GCO  860
C     EXAMINATION OF LINES FROM Z(I,J) TO IT'S NEIGHBORS.               GCO  870
C                                                                       GCO  880
      REAL XINT(4)                                                    GCO  890
C                                                                       GCO  900
C     XINT IS USED TO MARK INTERSECTIONS OF THE CONTOUR UNDER           GCO  910
C     CONSIDERATION WITH THE EDGES OF THE CELL BEING EXAMINED.          GCO  920
C                                                                       GCO  930
      REAL XY(2)                                                      GCO  940
C                                                                       GCO  950
C     XY IS USED TO COMPUTE COORDINATES FOR THE DRAW SUBROUTINE.        GCO  960
C                                                                       GCO  970
      EQUIVALENCE (L2(1),IMAX), (L2(2),JMAX), (L2(3),IMIN),           GCO  980
     *  (L2(4),JMIN)                                                  GCO  990
      EQUIVALENCE (IJ(1),I), (IJ(2),J)                                GCO 1000
      EQUIVALENCE (XY(1),X), (XY(2),Y)                                GCO 1010
C                                                                       GCO 1020
      DATA L1(3) /-1/, L1(4) /-1/                                     GCO 1030
      DATA I1 /1,0/, I2 /1,-1/, I3 /1,0,0,1,1,0/                      GCO 1040
C                                                                       GCO 1050
      L1(1) = NX                                                      GCO 1060
      L1(2) = NY                                                      GCO 1070
      DMAX = ZMAX                                                     GCO 1080
C                                                                       GCO 1090
C     SET THE CURRENT PEN POSITION.  THE DEFAULT POSITION CORRESPONDS   GCO 1100
C     TO Z(1,1).                                                        GCO 1110
C                                                                       GCO 1120
      X = 1.0                                                         GCO 1130
      Y = 1.0                                                         GCO 1140
      CALL DRAW(X, Y, 6)                                              GCO 1150
      ICUR = MAX0(1,MIN0(INT(X),NX))                                  GCO 1160
      JCUR = MAX0(1,MIN0(INT(Y),NY))                                  GCO 1170
C                                                                       GCO 1180
C     CLEAR THE BITMAP                                                  GCO 1190
C                                                                       GCO 1200
      CALL FILL0(BITMAP, 2*NX*NY*NCV)                                 GCO 1210
C                                                                       GCO 1220
C     SEARCH ALONG A RECTANGULAR SPIRAL PATH FOR A LINE SEGMENT HAVING  GCO 1230
C     THE FOLLOWING PROPERTIES:                                         GCO 1240
C          1.  THE END POINTS ARE NOT EXCLUDED,                         GCO 1250
C          2.  NO MARK HAS BEEN RECORDED FOR THE SEGMENT,               GCO 1260
C          3.  THE VALUES OF Z AT THE ENDS OF THE SEGMENT ARE SUCH THAT GCO 1270
C              ONE Z IS LESS THAN THE CURRENT CONTOUR VALUE, AND THE    GCO 1280
C              OTHER IS GREATER THAN OR EQUAL TO THE CURRENT CONTOUR    GCO 1290
C              VALUE.                                                   GCO 1300
C                                                                       GCO 1310
C     SEARCH ALL BOUNDARIES FIRST, THEN SEARCH INTERIOR LINE SEGMENTS.  GCO 1320
C     NOTE THAT THE INTERIOR LINE SEGMENTS NEAR EXCLUDED POINTS MAY BE  GCO 1330
C     BOUNDARIES.                                                       GCO 1340
C                                                                       GCO 1350
      IBKEY = 0                                                       GCO 1360
   10 I = ICUR                                                        GCO 1370
      J = JCUR                                                        GCO 1380
   20 IMAX = I                                                        GCO 1390
      IMIN = -I                                                       GCO 1400
      JMAX = J                                                        GCO 1410
      JMIN = -J                                                       GCO 1420
      IDIR = 0                                                        GCO 1430
C     DIRECTION ZERO IS +I, 1 IS +J, 2 IS -I, 3 IS -J.                  GCO 1440
```

```
   30 NXIDIR = IDIR + 1                                                    GCO 1450
      K = NXIDIR                                                           GCO 1460
      IF (NXIDIR.GT.3) NXIDIR = 0                                          GCO 1470
   40 I = IABS(I)                                                          GCO 1480
      J = IABS(J)                                                          GCO 1490
      IF (Z(I,J).GT.DMAX) GO TO 140                                        GCO 1500
      L = 1                                                                GCO 1510
C     L=1 MEANS HORIZONTAL LINE, L=2 MEANS VERTICAL LINE.                  GCO 1520
   50 IF (IJ(L).GE.L1(L)) GO TO 130                                        GCO 1530
      II = I + I1(L)                                                       GCO 1540
      JJ = J + I1(3-L)                                                     GCO 1550
      IF (Z(II,JJ).GT.DMAX) GO TO 130                                      GCO 1560
      ASSIGN 100 TO JUMP                                                   GCO 1570
C     THE NEXT 15 STATEMENTS (OR SO) DETECT BOUNDARIES.                    GCO 1580
   60 IX = 1                                                               GCO 1590
      IF (IJ(3-L).EQ.1) GO TO 80                                           GCO 1600
      II = I - I1(3-L)                                                     GCO 1610
      JJ = J - I1(L)                                                       GCO 1620
      IF (Z(II,JJ).GT.DMAX) GO TO 70                                       GCO 1630
      II = I + I2(L)                                                       GCO 1640
      JJ = J + I2(3-L)                                                     GCO 1650
      IF (Z(II,JJ).LT.DMAX) IX = 0                                         GCO 1660
   70 IF (IJ(3-L).GE.L1(3-L)) GO TO 90                                     GCO 1670
   80 II = I + I1(3-L)                                                     GCO 1680
      JJ = J + I1(L)                                                       GCO 1690
      IF (Z(II,JJ).GT.DMAX) GO TO 90                                       GCO 1700
      IF (Z(I+1,J+1).LT.DMAX) GO TO JUMP, (100, 280)                       GCO 1710
   90 IX = IX + 2                                                          GCO 1720
      GO TO JUMP, (100, 280)                                               GCO 1730
  100 IF (IX.EQ.3) GO TO 130                                               GCO 1740
      IF (IX+IBKEY.EQ.0) GO TO 130                                         GCO 1750
C     NOW DETERMINE WHETHER THE LINE SEGMENT IS CROSSED BY THE CONTOUR.    GCO 1760
      II = I + I1(L)                                                       GCO 1770
      JJ = J + I1(3-L)                                                     GCO 1780
      Z1 = Z(I,J)                                                          GCO 1790
      Z2 = Z(II,JJ)                                                        GCO 1800
      DO 120 ICV=1,NCV                                                     GCO 1810
        IF (IGET(BITMAP,2*(NX*(NY*(ICV-1)+J-1)+I-1)+L).NE.0) GO TO 120     GCO 1820
        IF (CV(ICV).LE.AMIN1(Z1,Z2)) GO TO 110                             GCO 1830
        IF (CV(ICV).LE.AMAX1(Z1,Z2)) GO TO 190                             GCO 1840
  110   CALL MARK1(BITMAP, 2*(NX*(NY*(ICV-1)+J-1)+I-1)+L)                  GCO 1850
  120 CONTINUE                                                             GCO 1860
  130 L = L + 1                                                            GCO 1870
      IF (L.LE.2) GO TO 50                                                 GCO 1880
  140 L = MOD(IDIR,2) + 1                                                  GCO 1890
      IJ(L) = ISIGN(IJ(L),L1(K))                                           GCO 1900
C                                                                          GCO 1910
C     LINES FROM Z(I,J) TO Z(I+1,J) AND Z(I,J+1) ARE NOT SATISFACTORY.     GCO 1920
C     CONTINUE THE SPIRAL.                                                 GCO 1930
C                                                                          GCO 1940
  150 IF (IJ(L).GE.L1(K)) GO TO 170                                        GCO 1950
      IJ(L) = IJ(L) + 1                                                    GCO 1960
      IF (IJ(L).GT.L2(K)) GO TO 160                                        GCO 1970
      GO TO 40                                                             GCO 1980
  160 L2(K) = IJ(L)                                                        GCO 1990
      IDIR = NXIDIR                                                        GCO 2000
      GO TO 30                                                             GCO 2010
  170 IF (IDIR.EQ.NXIDIR) GO TO 180                                        GCO 2020
      NXIDIR = NXIDIR + 1                                                  GCO 2030
      IJ(L) = L1(K)                                                        GCO 2040
      K = NXIDIR                                                           GCO 2050
      L = 3 - L                                                            GCO 2060
      IJ(L) = L2(K)                                                        GCO 2070
      IF (NXIDIR.GT.3) NXIDIR = 0                                          GCO 2080
      GO TO 150                                                            GCO 2090
  180 IF (IBKEY.NE.0) RETURN                                               GCO 2100
      IBKEY = 1                                                            GCO 2110
      GO TO 10                                                             GCO 2120
C                                                                          GCO 2130
C     AN ACCEPTABLE LINE SEGMENT HAS BEEN FOUND.                           GCO 2140
C     FOLLOW THE CONTOUR UNTIL IT EITHER HITS A BOUNDARY OR CLOSES.        GCO 2150
C                                                                          GCO 2160
  190 IEDGE = L                                                            GCO 2170
      CVAL = CV(ICV)                                                       GCO 2180
      IF (IX.NE.1) IEDGE = IEDGE + 2                                       GCO 2190
      IFLAG = 2 + IBKEY                                                    GCO 2200
```

```
      XINT(IEDGE) = (CVAL-Z1)/(Z2-Z1)                                    GCO 2210
  200 XY(L) = FLOAT(IJ(L)) + XINT(IEDGE)                                 GCO 2220
      XY(3-L) = FLOAT(IJ(3-L))                                           GCO 2230
      CALL MARK1(BITMAP, 2*(NX*(NY*(ICV-1)+J-1)+I-1)+L)                  GCO 2240
      CALL DRAW(X, Y, IFLAG+10*ICV)                                      GCO 2250
      IF (IFLAG.LT.4) GO TO 210                                          GCO 2260
      ICUR = I                                                           GCO 2270
      JCUR = J                                                           GCO 2280
      GO TO 20                                                           GCO 2290
C                                                                        GCO 2300
C     CONTINUE A CONTOUR.  THE EDGES ARE NUMBERED CLOCKWISE WITH         GCO 2310
C     THE BOTTOM EDGE BEING EDGE NUMBER ONE.                             GCO 2320
C                                                                        GCO 2330
  210 NI = 1                                                             GCO 2340
      IF (IEDGE.LT.3) GO TO 220                                          GCO 2350
      I = I - I3(IEDGE)                                                  GCO 2360
      J = J - I3(IEDGE+2)                                                GCO 2370
  220 DO 250 K=1,4                                                       GCO 2380
        IF (K.EQ.IEDGE) GO TO 250                                        GCO 2390
        II = I + I3(K)                                                   GCO 2400
        JJ = J + I3(K+1)                                                 GCO 2410
        Z1 = Z(II,JJ)                                                    GCO 2420
        II = I + I3(K+1)                                                 GCO 2430
        JJ = J + I3(K+2)                                                 GCO 2440
        Z2 = Z(II,JJ)                                                    GCO 2450
        IF (CVAL.LE.AMIN1(Z1,Z2)) GO TO 250                              GCO 2460
        IF (CVAL.GT.AMAX1(Z1,Z2)) GO TO 250                              GCO 2470
        IF (K.EQ.1) GO TO 230                                            GCO 2480
        IF (K.NE.4) GO TO 240                                            GCO 2490
  230   ZZ = Z1                                                          GCO 2500
        Z1 = Z2                                                          GCO 2510
        Z2 = ZZ                                                          GCO 2520
  240   XINT(K) = (CVAL-Z1)/(Z2-Z1)                                      GCO 2530
        NI = NI + 1                                                      GCO 2540
        KS = K                                                           GCO 2550
  250 CONTINUE                                                           GCO 2560
      IF (NI.EQ.2) GO TO 260                                             GCO 2570
C                                                                        GCO 2580
C     THE CONTOUR CROSSES ALL FOUR EDGES OF THE CELL BEING EXAMINED.     GCO 2590
C     CHOOSE THE LINES TOP-TO-LEFT AND BOTTOM-TO-RIGHT IF THE            GCO 2600
C     INTERPOLATION POINT ON THE TOP EDGE IS LESS THAN THE INTERPOLATIONGCO 2610
C     POINT ON THE BOTTOM EDGE.  OTHERWISE, CHOOSE THE OTHER PAIR.  THISGCO 2620
C     METHOD PRODUCES THE SAME RESULTS IF THE AXES ARE REVERSED.  THE    GCO 2630
C     CONTOUR MAY CLOSE AT ANY EDGE, BUT MUST NOT CROSS ITSELF INSIDE    GCO 2640
C     ANY CELL.                                                          GCO 2650
C                                                                        GCO 2660
      KS = 5 - IEDGE                                                     GCO 2670
      IF (XINT(3).LT.XINT(1)) GO TO 260                                  GCO 2680
      KS = 3 - IEDGE                                                     GCO 2690
      IF (KS.LE.0) KS = KS + 4                                           GCO 2700
C                                                                        GCO 2710
C     DETERMINE WHETHER THE CONTOUR WILL CLOSE OR RUN INTO A BOUNDARY    GCO 2720
C     AT EDGE KS OF THE CURRENT CELL.                                    GCO 2730
C                                                                        GCO 2740
  260 L = KS                                                             GCO 2750
      IFLAG = 1                                                          GCO 2760
      ASSIGN 280 TO JUMP                                                 GCO 2770
      IF (KS.LT.3) GO TO 270                                             GCO 2780
      I = I + I3(KS)                                                     GCO 2790
      J = J + I3(KS+2)                                                   GCO 2800
      L = KS - 2                                                         GCO 2810
  270 IF (IGET(BITMAP,2*(NX*(NY*(ICV-1)+J-1)+I-1)+L).EQ.0) GO TO 60      GCO 2820
      IFLAG = 5                                                          GCO 2830
      GO TO 290                                                          GCO 2840
  280 IF (IX.NE.0) IFLAG = 4                                             GCO 2850
  290 IEDGE = KS + 2                                                     GCO 2860
      IF (IEDGE.GT.4) IEDGE = IEDGE - 4                                  GCO 2870
      XINT(IEDGE) = XINT(KS)                                             GCO 2880
      GO TO 200                                                          GCO 2890
C                                                                        GCO 2900
      END                                                                GCO 2910


      DIMENSION Z(51,51), C(10), WORK(1680)                              MAN   10
C     DIMENSION OF WORK IS LARGE ENOUGH TO CONTAIN                       MAN   20
C     2*(DIMENSION OF C)*(TOTAL DIMENSION OF Z) USEFUL BITS.  SEE THE    MAN   30
C     BITMAP ROUTINES ACCESSED BY GCONTR.                                MAN   40
```

```
      REAL MU                                                          MAN  50
      EXTERNAL DRAW                                                    MAN  60
      COMMON /CUR/ XCUR, YCUR                                          MAN  70
      DATA C(1), C(2), C(3), C(4), C(5) /3.05,3.2,3.5,3.50135,3.6/     MAN  80
      DATA C(6), C(7), C(8), C(9), C(10) /3.766413,4.0,4.130149,5.0,   MAN  90
     * 10.0/                                                           MAN 100
      DATA NX /51/, NY /51/, NF /10/                                   MAN 110
      DATA XMIN /-2.0/, XMAX /2.0/, YMIN /-2.0/, YMAX /2.0/, MU /0.3/  MAN 120
      DX = (XMAX-XMIN)/FLOAT(NX-1)                                     MAN 130
      DY = (YMAX-YMIN)/FLOAT(NY-1)                                     MAN 140
      XCUR = 1.0                                                       MAN 150
      YCUR = 1.0                                                       MAN 160
      IF (MOD(NX,2).NE.0) YCUR = FLOAT(NY)                             MAN 170
      IF (MOD(NY,2).NE.0) XCUR = FLOAT(NX)                             MAN 180
      X = XMIN - DX                                                    MAN 190
      DO 20 I=1,NX                                                     MAN 200
        Y = YMIN - DY                                                  MAN 210
        X = X + DX                                                     MAN 220
        DO 10 J=1,NY                                                   MAN 230
          Y = Y + DY                                                   MAN 240
          Z(I,J) = (1.0-MU)*(2.0/SQRT((X-MU)**2+Y**2)+(X-MU)**2+Y**2)  MAN 250
     *       + MU*(2.0/SQRT((X+1.0-MU)**2+Y**2)+(X+1.0-MU)**2+Y**2)    MAN 260
   10   CONTINUE                                                       MAN 270
   20 CONTINUE                                                       MAN 280
      CALL GCONTR(Z, 51, NX, NY, C, NF, 1.E6, WORK, DRAW)              MAN 290
      STOP                                                             MAN 300
      END                                                              MAN 310

      REAL Z(51,51), C(10), CVAL(10), MU                               MAN  10
      INTEGER WORK(1680), L(10), CLAB(10)                              MAN  20
C     DIMENSION OF WORK IS LARGE ENOUGH TO CONTAIN                     MAN  30
C     2*(DIMENSION OF C)*(TOTAL DIMENSION OF Z) USEFUL BITS.  SEE THE  MAN  40
C     BITMAP ROUTINES ACCESSED BY GCONTR.                              MAN  50
      EXTERNAL DRAW                                                    MAN  60
      COMMON /GCTCOM/ XCUR, YCUR, XL, YL, CVAL, CLAB, NCH              MAN  70
      DATA C(1), C(2), C(3), C(4), C(5) /3.05,3.2,3.5,3.50135,3.6/     MAN  80
      DATA C(6), C(7), C(8), C(9), C(10) /3.766413,4.0,4.130149,5.0,   MAN  90
     * 10.0/                                                           MAN 100
      DATA L(1), L(2), L(3), L(4), L(5) /1HA,1HB,1HC,1HD,1HE/          MAN 110
      DATA L(6), L(7), L(8), L(9), L(10) /1HF,1HG,1HH,1HI,1HJ/         MAN 120
      DATA NX /51/, NY /51/, NF /10/, NXG /5/, NYG /5/                 MAN 130
      DATA XMIN /-2.0/, XMAX /2.0/, YMIN /-2.0/, YMAX /2.0/, MU /0.3/  MAN 140
      DATA XLEN /8.0/, YLEN /8.0/                                      MAN 150
C     INITIALIZE PLOTTING SUBROUTINES.                                 MAN 160
      CALL PLOTS                                                       MAN 170
      DX = (XMAX-XMIN)/FLOAT(NX-1)                                     MAN 180
      DY = (YMAX-YMIN)/FLOAT(NY-1)                                     MAN 190
      XL = XLEN/FLOAT(NX)                                              MAN 200
      YL = YLEN/FLOAT(NY)                                              MAN 210
      XCUR = 1.0                                                       MAN 220
      YCUR = 1.0                                                       MAN 230
      IF (MOD(NX,2).NE.0) YCUR = FLOAT(NY)                             MAN 240
      IF (MOD(NY,2).NE.0) XCUR = FLOAT(NX)                             MAN 250
      X = XMIN - DX                                                    MAN 260
      DO 20 I=1,NX                                                     MAN 270
        Y = YMIN - DY                                                  MAN 280
        X = X + DX                                                     MAN 290
        DO 10 J=1,NY                                                   MAN 300
          Y = Y + DY                                                   MAN 310
C     EVALUATE FUNCTION TO BE PLOTTED.                                 MAN 320
          Z(I,J) = (1.0-MU)*(2.0/SQRT((X-MU)**2+Y**2)+(X-MU)**2+Y**2)  MAN 330
     *       + MU*(2.0/SQRT((X+1.0-MU)**2+Y**2)+(X+1.0-MU)**2+Y**2)    MAN 340
   10   CONTINUE                                                       MAN 350
   20 CONTINUE                                                       MAN 360
      DO 30 I=1,NF                                                     MAN 370
        CVAL(I) = C(I)                                                 MAN 380
        CLAB(I) = L(I)                                                 MAN 390
   30 CONTINUE                                                       MAN 400
      NCH = 1                                                          MAN 410
C     PEN UP MOVE TO BELOW LOWER LEFT CORNER OF PAGE.                  MAN 420
C     THIS CALL WORKS DIFFERENTLY ON DIFFERENT MACHINES.  YOU MAY      MAN 430
C     NEED TO CHANGE IT.                                               MAN 440
      CALL PLOT(0.0, -11.0, -3)                                        MAN 450
C     PEN UP MOVE TO 1 INCH ABOVE LOWER LEFT CORNER OF PAGE.           MAN 460
      CALL PLOT(0.0, 1.0, -3)                                          MAN 470
```

```
      SX = 8.0/FLOAT(NXG)                                                MAN  480
      SY = 8.0/FLOAT(NXG)                                                MAN  490
C     DRAW A GRID.                                                       MAN  500
      CALL CGRID(1, NXG, SX, 0.0, 0.0, NYG, SY, 0.0, 0.0)                MAN  510
C     DRAW THE CONTOUR PLOTS.                                            MAN  520
      CALL GCONTR(Z, 51, NX, NY, CVAL, NF, 1.0E6, WORK, DRAW)            MAN  530
      XX = 9.0                                                           MAN  540
      YY = 8.0                                                           MAN  550
C     WRITE A TABLE OF CONTOUR LABELS AND VALUES.                        MAN  560
      CALL SYMBOL(XX, YY+0.14, 0.07, 10HCONTOUR ID, 0.0, 10)             MAN  570
      DO 40 I=1,NF                                                       MAN  580
        CALL SYMBOL(XX, YY, 0.07, L(I), 0.0, 2)                          MAN  590
        CALL NUMBER(XX+0.12, YY, 0.07, C(I), 0.0, 5)                     MAN  600
        YY = YY - 0.14                                                   MAN  610
   40 CONTINUE                                                           MAN  620
C     PEN UP MOVE TO BELOW LOWER RIGHT CORNER OF PAGE.                   MAN  630
C     THIS CALL WORKS DIFFERENTLY ON DIFFERENT MACHINES.  YOU MAY NEED   MAN  640
C     TO CHANGE IT, OR YOU MAY NOT NEED IT.                              MAN  650
      CALL PLOT(10.0, -11.0, -3)                                         MAN  660
C     REDUCE PICTURE SIZE, PLOT END OF FILE INFORMATION.                 MAN  670
C     THE END OF FILE INFORMATION MAY NOT BE AVAILABLE AT ALL SITES.     MAN  680
C     IF NOT AVAILABLE, CHANGE THE NEXT TWO STATEMENTS TO COMMENTS.      MAN  690
      CALL FACTOR(0.3)                                                   MAN  700
      CALL PLOT(0.0, 0.0, 999)                                           MAN  710
      STOP                                                               MAN  720
C                                                                        MAN  730
      END                                                                MAN  740


      SUBROUTINE DRAW(X, Y, IFLAG)                                       DRA   10
C     THIS SUBROUTINE USES CALCOMP PLOT ROUTINES TO DRAW LINES FOR THE   DRA   20
C     CONTOUR PLOTTING ROUTINE GCONTR.                                   DRA   30
      REAL CVAL(10)                                                      DRA   40
      INTEGER CLAB(10)                                                   DRA   50
      COMMON /GCTCOM/ XCUR, YCUR, XL, YL, CVAL, CLAB, NCH                DRA   60
      DATA IBLANK /1H /                                                  DRA   70
      IH = IFLAG/10                                                      DRA   80
      IL = IFLAG - 10*IH                                                 DRA   90
      IF (IL.EQ.6) GO TO 40                                              DRA  100
      IPEN = 2                                                           DRA  110
      IF (IL.EQ.2) IPEN = 3                                              DRA  120
      IF (IL.EQ.3) IPEN = 3                                              DRA  130
      XCUR = X                                                           DRA  140
      YCUR = Y                                                           DRA  150
      XX = (X-1.0)*XL                                                    DRA  160
      YY = (Y-1.0)*YL                                                    DRA  170
      CALL PLOT(XX, YY, IPEN)                                            DRA  180
      IF (IL.LT.2) GO TO 30                                              DRA  190
      IF (IL.GT.4) GO TO 30                                              DRA  200
      IF (NCH.LT.1) GO TO 30                                             DRA  210
      IF (CLAB(IH).EQ.IBLANK) GO TO 30                                   DRA  220
      IF (CLAB(IH).NE.0) GO TO 10                                        DRA  230
      CALL NUMBER(XX, YY-0.03, 0.07, CVAL(IH), 0.0, -1)                  DRA  240
      GO TO 20                                                           DRA  250
   10 CALL SYMBOL(XX, YY-0.03, 0.07, CLAB(IH), 0.0, NCH)                 DRA  260
   20 CALL PLOT(XX, YY, 3)                                               DRA  270
   30 RETURN                                                             DRA  280
   40 X = XCUR                                                           DRA  290
      Y = YCUR                                                           DRA  300
      RETURN                                                             DRA  310
C                                                                        DRA  320
      END                                                                DRA  330


      SUBROUTINE DRAW(X, Y, IFLAG)                                       DRA   10
C                                                                        DRA   20
C     DO OUTPUT FOR GCONTR.                                              DRA   30
C                                                                        DRA   40
      INTEGER PRINT                                                      DRA   50
      COMMON /CUR/ XCUR, YCUR                                            DRA   60
      DATA PRINT /6/                                                     DRA   70
C     PRINT IS THE SYSTEM PRINTER FORTRAN I/O UNIT NUMBER.               DRA   80
      ICONT = IFLAG/10                                                   DRA   90
      JUMP = MOD(IFLAG,10)                                               DRA  100
      GO TO (10, 20, 30, 40, 50, 60), JUMP                               DRA  110
   10 WRITE (PRINT,99999) ICONT, X, Y                                    DRA  120
      GO TO 70                                                           DRA  130
```

```
   20 WRITE (PRINT,99998) ICONT, X, Y                                   DRA  140
      GO TO 70                                                          DRA  150
   30 WRITE (PRINT,99997) ICONT, X, Y                                   DRA  160
      GO TO 70                                                          DRA  170
   40 WRITE (PRINT,99996) ICONT, X, Y                                   DRA  180
      GO TO 70                                                          DRA  190
   50 WRITE (PRINT,99995) ICONT, X, Y                                   DRA  200
      GO TO 70                                                          DRA  210
   60 WRITE (PRINT,99994)                                               DRA  220
      X = XCUR                                                          DRA  230
      Y = YCUR                                                          DRA  240
   70 RETURN                                                            DRA  250
99999 FORMAT (17H CONTINUE CONTOUR, I3, 3H TO, 1P2E14.7)                DRA  260
99998 FORMAT (14H START CONTOUR, I3, 19H ON THE BOUNDARY AT, 1P2E14.7)   DRA  270
99997 FORMAT (14H START CONTOUR, I3, 19H IN THE INTERIOR AT, 1P2E14.7)   DRA  280
99996 FORMAT (15H FINISH CONTOUR, I3, 19H ON THE BOUNDARY AT, 1P2E14.7)  DRA  290
99995 FORMAT (15H FINISH CONTOUR, I3, 19H IN THE INTERIOR AT, 1P2E14.7)  DRA  300
99994 FORMAT (33H REQUEST FOR CURRENT PEN POSITION)                      DRA  310
      END                                                               DRA  320

      SUBROUTINE CGRID(NOPT, NX, SX, XS, XF, NY, SY, YS, YF)            CGR   10
C                                                                       CGR   20
C SUBROUTINE WHICH DRAWS A FRAME AROUND THE PLOT AND DRAWS              CGR   30
C EITHER TICK MARKS OR GRID LINES.                                      CGR   40
C                                                                       CGR   50
C PARAMETERS:  NOPT -- =0, DRAW TICKS ONLY                              CGR   60
C                      =1, DRAW GRID LINES                              CGR   70
C                      =2, DRAW GRID LINES TO EDGE OF FRAME.            CGR   80
C              NX -- NUMBER OF INTERVALS IN X DIRECTION                 CGR   90
C              SX -- SPACING IN INCHES BETWEEN TICK MARKS OR GRID LINES CGR  100
C                    ALONG THE X AXIS                                   CGR  110
C              XS -- LOCATION OF FIRST TICK OR GRID LINE ON X AXIS      CGR  120
C              XF -- LOCATION OF RIGHT EDGE OF FRAME                    CGR  130
C              NY -- NUMBER OF INTERVALS IN Y DIRECTION                 CGR  140
C              SY -- SPACING IN INCHES BETWEEN TICK MARKS OR GRID LINES CGR  150
C                    ALONG THE Y AXIS                                   CGR  160
C              YS -- LOCATION OF FIRST TICK OR GRID LINE ON Y AXIS      CGR  170
C              YF -- LOCATION OF TOP EDGE OF FRAME                      CGR  180
C ASSUMPTIONS: NX, SX, NY, SY ALL POSITIVE.                             CGR  190
C              THE LOWER LEFT-HAND CORNER OF THE FRAME IS DRAWN AT (0,0)CGR  200
C              IF XS10, USE 0; IF YS10, USE 0                           CGR  210
C              IF XF1=0, USE NX*SX; IF YF1=0, USE NY*SY.                CGR  220
C                                                                       CGR  230
      XINC = SX                                                         CGR  240
      YINC = SY                                                         CGR  250
      XLGTH = FLOAT(NX)*SX                                              CGR  260
      YLGTH = FLOAT(NY)*SY                                              CGR  270
      XMIN = AMAX1(XS,0.0)                                              CGR  280
      YMIN = AMAX1(YS,0.0)                                              CGR  290
      XMAX = AMAX1(XF,XLGTH+XMIN)                                       CGR  300
      YMAX = AMAX1(YF,YLGTH+YMIN)                                       CGR  310
C                                                                       CGR  320
C     DRAW FRAME.                                                       CGR  330
C                                                                       CGR  340
      CALL PLOT(0.0, 0.0, 3)                                            CGR  350
      CALL PLOT(XMAX, 0.0, 2)                                           CGR  360
      CALL PLOT(XMAX, YMAX, 2)                                          CGR  370
      CALL PLOT(0.0, YMAX, 2)                                           CGR  380
      CALL PLOT(0.0, 0.0, 2)                                            CGR  390
      IF (NOPT.NE.0) GO TO 130                                          CGR  400
C                                                                       CGR  410
C     DRAW TICK MARKS.                                                  CGR  420
C                                                                       CGR  430
      DO 120 J=1,4                                                      CGR  440
        GO TO (10, 50, 20, 40), J                                       CGR  450
   10   X2 = 0.0                                                        CGR  460
        IF (XMIN.NE.0.0) X2 = XMIN - SX                                 CGR  470
        Y2 = 0.0                                                        CGR  480
        GO TO 30                                                        CGR  490
   20   XINC = -SX                                                      CGR  500
        X2 = XMIN + XLGTH + SX                                          CGR  510
        IF (XMAX.EQ.XMIN+XLGTH) X2 = XMAX                               CGR  520
        Y2 = YMAX                                                       CGR  530
   30   Y1 = Y2                                                         CGR  540
        Y2 = Y2 + SIGN(0.125,XINC)                                      CGR  550
```

```
      N = NX                                                          CGR  560
      IF (ABS(XMAX-XMIN-XLGTH)+ABS(XMIN)) 70, 80, 70                  CGR  570
 40   YINC = -SY                                                      CGR  580
      Y2 = YMIN + YLGTH + SY                                          CGR  590
      IF (YMAX.EQ.YMIN+YLGTH) Y2 = YMAX                               CGR  600
      X2 = 0.0                                                        CGR  610
      GO TO 60                                                        CGR  620
 50   Y2 = 0.0                                                        CGR  630
      IF (YMIN.NE.0.0) Y2 = YMIN - SY                                 CGR  640
      X2 = XMAX                                                       CGR  650
 60   X1 = X2                                                         CGR  660
      N = NY                                                          CGR  670
      X2 = X2 - SIGN(0.125,YINC)                                      CGR  680
      IF (ABS(YMAX-YMIN-YLGTH)+ABS(YMIN)) 70, 80, 70                  CGR  690
 70   N = N + 1                                                       CGR  700
 80   DO 110 I=1,N                                                    CGR  710
        IF (MOD(J,2).EQ.0) GO TO 90                                   CGR  720
        X2 = X2 + XINC                                                CGR  730
        X1 = X2                                                       CGR  740
        GO TO 100                                                     CGR  750
 90     Y2 = Y2 + YINC                                                CGR  760
        Y1 = Y2                                                       CGR  770
100     CALL PLOT(X1, Y1, 3)                                          CGR  780
        CALL PLOT(X2, Y2, 2)                                          CGR  790
110   CONTINUE                                                        CGR  800
120 CONTINUE                                                          CGR  810
    GO TO 240                                                         CGR  820
C                                                                     CGR  830
C     DRAW GRID LINES                                                 CGR  840
C                                                                     CGR  850
130 X1 = XMIN                                                         CGR  860
    X2 = XMIN + XLGTH                                                 CGR  870
    IF (NOPT.NE.2) GO TO 140                                          CGR  880
    X1 = 0.0                                                          CGR  890
    X2 = XMAX                                                         CGR  900
140 Y1 = YMIN - SY                                                    CGR  910
    N = NY + 1                                                        CGR  920
    IF (YMAX.EQ.YMIN+YLGTH) N = N - 1                                 CGR  930
    IF (YMIN.NE.0.0) GO TO 150                                        CGR  940
    Y1 = 0.0                                                          CGR  950
    N = N - 1                                                         CGR  960
150 IF (N.LE.0) GO TO 170                                             CGR  970
    J = 1                                                             CGR  980
    DO 160 I=1,N                                                      CGR  990
      J = -J                                                          CGR 1000
      Y1 = Y1 + SY                                                    CGR 1010
      CALL PLOT(X1, Y1, 3)                                            CGR 1020
      CALL PLOT(X2, Y1, 2)                                            CGR 1030
      XX = X1                                                         CGR 1040
      X1 = X2                                                         CGR 1050
      X2 = XX                                                         CGR 1060
160 CONTINUE                                                          CGR 1070
170 Y1 = YMIN + YLGTH                                                 CGR 1080
    Y2 = YMIN                                                         CGR 1090
    IF (NOPT.NE.2) GO TO 180                                          CGR 1100
    Y1 = YMAX                                                         CGR 1110
    Y2 = 0.0                                                          CGR 1120
180 N = NX + 1                                                        CGR 1130
    IF (J.LT.0) GO TO 200                                             CGR 1140
    X1 = XMIN - SX                                                    CGR 1150
    IF (XMAX.EQ.XMIN+XLGTH) N = N - 1                                 CGR 1160
    IF (XMIN.NE.0.0) GO TO 190                                        CGR 1170
    X1 = 0.0                                                          CGR 1180
    N = N - 1                                                         CGR 1190
190 IF (N.LE.0) GO TO 240                                             CGR 1200
    XINC = SX                                                         CGR 1210
    GO TO 220                                                         CGR 1220
200 X1 = XMIN + XLGTH + SX                                            CGR 1230
    IF (XMIN.EQ.0.0) N = N - 1                                        CGR 1240
    IF (XMAX.NE.XLGTH+XMIN) GO TO 210                                 CGR 1250
    N = N - 1                                                         CGR 1260
    X1 = XMAX                                                         CGR 1270
210 XINC = -SX                                                        CGR 1280
220 DO 230 I=1,N                                                      CGR 1290
      X1 = X1 + XINC                                                  CGR 1300
      CALL PLOT(X1, Y1, 3)                                            CGR 1310
```

```
            CALL PLOT(X1, Y2, 2)                                    CGR 1320
            XX = Y1                                                 CGR 1330
            Y1 = Y2                                                 CGR 1340
            Y2 = XX                                                 CGR 1350
      230 CONTINUE                                                  CGR 1360
      240 RETURN                                                    CGR 1370
C                                                                   CGR 1380
          END                                                       CGR 1390
```

ALGORITHM 532
Software for Roundoff Analysis [Z]

WEBB MILLER
University of California, Santa Barbara
and
DAVID SPOONER
Pennsylvania State University

Key Words and Phrases: automatic roundoff analysis, numerical stability, numerical linear algebra
CR Categories: 5.10, 5.11, 5.14
Language: Fortran

DESCRIPTION

This software package is a complement to [1] where its usage and performance are described.

REFERENCES

1. MILLER, W., AND SPOONER, D. Software for roundoff analysis, II. *ACM Trans. Math. Software 4*, 4 (Dec. 1978), 369–387.

ALGORITHM

NAME(n): indicates a Fortran module with n records
NAME^D: indicates test data for the minicompiler
Contents for the minicompiler: DATA(570), MAIN(388), CODGEN(445), NEXT(12), CODOPT(148), INSERT(69), GETNAM(32), GETDIM(34), RDIM(51), ADD(114), STORE(69), GETVAL(88), SYMINT(39), LEXAN(422), GETSTM(343), KFIND(118), INTERP(322), REALOP(25), OPER(504), FINISH(203), ADDTMP(70), ADDSUB(51), ADDNAM(47), TERROR(58), ERROR(80), IERROR(95)

```
C                 *************************
C                 *                       *
C                 *       OVERVIEW        *
C                 *                       *
C                 *************************
C
C
C  THIS PROGRAM IS A COMPILER WHICH TAKES AS DATA A PROGRAM WRITTEN
C  IN A SIMPLE PROGRAMMING LANGUAGE ( A DESCRIPTION OF WHICH FOLLOWS)
C  AND PRODUCES AS OUTPUT A TRANSLATION OF THAT PROGRAM INTO A SERIES
C  OF ASSIGNMENT STATEMENTS.  THE STRAIGHT LINE CODE TRANSLATION IS
```

Received 21 August 1975 and 4 March 1977.

This work was supported in part by NSF Grants GJ-42968 and MCS 76-13561A01.
Authors' addresses: W. Miller, Department of Mathematics, University of California, Santa Barbara, Santa Barbara, CA 93106; D. Spooner, Department of Computer Science, 303 Whitmore Laboratory, Pennsylvania State University, University Park, PA 16802.

```
C  PRINTED IN A READABLE FORM FOR THE COMPILER USER AND ALSO PUNCHED
C  IN  A CODED FORM FOR USE AS INPUT TO ANY OF SEVERAL ROUNDOFF ERROR
C  ANALYSIS PROGRAMS OF WEBB MILLER.
C  THE MAIN ROUTINE OF THE COMPILER IS AN LL-1 PARSING ROUTINE.  IT
C  CALLS THE LEXICAL ANALYZER TO PRODUCE INTERNAL TOKENS BY SCANNING
C  THE SOURCE CODE.  THE PARSER ALSO CALLS THE ROUTINE CODGEN WHICH
C  PRODUCES INTERMEDIATE CODE FOR THE INTERPRETER AND WHICH PERFORMS
C  SOME SYNTAX CHECKING AND OTHER PERIPHERAL FUNCTIONS NECESSARY
C  FOR PARSING.
C  WHEN THE ENTIRE INPUT HAS BEEN SCANNED AND PARSED, THE INTERPRETING
C  ROUTINES ARE ACTIVATED.  THESE INTERPRET THE INTERMEDIATE CODE AND
C  GENERATE THE FINAL STRAIGHT LINE PRINTED AND PUNCHED OUTPUT.  DURING
C  INTERPRETATION, ALL INTEGER EXPRESSIONS ARE ACTUALLY EVALUATED IN
C  ORDER TO PERFORM THE CORRECT NUMBER OF ITERATIONS OF EXPLICIT FOR-
C  LOOPS AND OF FOR-LOOPS IMPLICIT IN SUMMATION EXPRESSIONS, AND TO
C  INTERPRETIVELY PERFORM IF-THEN TESTS.  IN CONTRAST, NO ACTUAL
C  REAL ARITHMETIC COMPUTATION IS DONE.  THROUGHOUT, ALL REAL
C  VARIABLES ARE TREATED SYMBOLICALLY AS BEING THE N-TH INPUT VALUE,
C  INTERMEDIATE VALUE, OR REAL CONSTANT.
C  BOTH THE LEXICAL ANALYZER AND THE INTERPRETER COMPRISE SEVERAL SUB-
C  ROUTINES.IN ADDITION THERE ARE COLLECTIONS OF SYMBOL TABLE ROUTINES,
C  ERROR ROUTINES AND ROUTINES TO PRINT AND PUNCH THE COMPILER'S
C  OUTPUT.
C  IF ANY ERRORS ARE ENCOUNTERED DURING PARSING, THE PARSER CONTINUES
C  TO CHECK FOR SYNTAX ERRORS, BUT NO FURTHER INTERMEDIATE CODE WILL BE
C  GENERATED, AND NO INTERPRETATION WILL TAKE PLACE.  SIMILARLY, IF
C  EXECUTION ERRORS ARE IDENTIFIED DURING INTERPRETATION, NO STRAIGHT
C  LINE CODE WILL BE EITHER PRINTED OR PUNCHED.
C
C
C            ****************************************
C            *                                      *
C            *        THE SOURCE LANGUAGE           *
C            *                                      *
C            ****************************************
C
C
C       THE LANGUAGE TO BE COMPILED IS A SIMPLE LANGUAGE DESIGNED FOR
C  CODING NUMERICAL ALGORITHMS.  IT BASICALLY INCLUDES REAL ASSIGN-
C  MENT STATEMENTS, DIMENSION STATEMENTS AND SOME BLOCK STRUCTURE
C  IMPOSED BY FOR-LOOPS, AND IF-THEN TESTS. THERE ARE NO MIXED-MODE
C  ARITHMETIC EXPRESSIONS, AND NO STATEMENT LABELS.  INTEGER EXPRESSIONS
C  AND VARIABLES ARE USED ONLY FOR DIMENSIONING REAL ARRAYS, FOR
C  DEFINING BOUNDS IN FOR-LOOPS AND SUMMATION EXPRESSION LOOPS, AND FOR
C  VARIABLES TO BE TESTED IN IF-THEN STATEMENTS.
C         THE CARD FORMAT IS SIMILAR TO FORTRAN.  STATEMENTS APPEAR
C  IN COLUMNS 7-72, AND A 1 IN COLUMN 6 INDICATES CONTINUATION.  A
C  C IN COLUMN 1 INDICATES A COMMENT.
C         THERE ARE SEVEN STATEMENT TYPES, BRIEFLY DESCRIBED BELOW:
C
C  1. THE TEST STATEMENT
C
C     USE:  TO ASSIGN VALUES TO INTEGER VARIABLES WHICH WILL THEMSELVES
C           BE USED TO DIMENSION REAL ARRAYS.
C
C     FORM: TEST(I1,I2,I3,...)
C
C           WHERE EACH I IS AN ASSIGNMENT STATEMENT OF THE FORM
C           INTEGER VARIABLE = INTEGER CONSTANT.
C
C  2. THE DIMENSION STATEMENT
C
C     USE:  TO ASSIGN DIMENSIONS TO ARRAYS.  THE ARRAY NAME MAY BE
C           EITHER A DEFAULT REAL OR INTEGER IDENTIFIER.  ITS USE IN
C           THE DIMENSION STATEMENT CONSTITUTES AN IMPLICIT REAL
C           DECLARATION.
C
C     FORM: DIMENSION(D1,D2,D3 ..)
C
C           WHERE EACH D IS OF ONE OF THE FOLLOWING FORMS:
C           IDENTIFIER(I)
C           IDENTIFIER(I1,I2)
C           AND EACH I IS AN INTEGER VARIABLE OR INTEGER CONSTANT.
C
C  NOTE: DIMENSION AND TEST STATEMENTS ARE NON-EXECUTABLE AND MUST
C        PRECEED ALL EXECUTABLE STATEMENTS.  IN ADDITION, AN INTEGER
```

```
C        VARIABLE APPEARING IN A DIMENSION STATEMENT MUST BE ASSIGNED A
C        VALUE IN A PRECEEDING TEST STATEMENT.
C
C  3.  THE INPUT/OUTPUT STATEMENTS
C
C     USE:  TO NOTIFY THE COMPILER THAT CERTAIN VALUES WILL BE SUPPLIED
C           BY THE PROGRAMMER AS INITIAL VALUES FOR REAL VARIABLES WHEN
C           THE STRAIGHT LINE CODE IS USED AS INPUT FOR A ROUNDOFF ERROR
C           ANALYSIS; OR THAT CERTAIN REAL VARIABLES ARE EXPECTED TO
C           RECEIVE VALUES AS A RESULT OF RUNNING THE PROGRAM BEING
C           COMPILED.
C
C     FORM: INPUT(D1,D2,D3,...)
C           OUTPUT(D1,D2,D3,..)
C
C           EACH D IS EITHER THE NAME OF A REAL SCALAR, A SINGLE ARRAY
C           ELEMENT, OR AN ENTIRE ARRAY.  IN THE LATTER CASE THE ARRAY
C           WILL BE INPUT(OUTPUT) IN COLUMN MAJOR ORDER.
C           NOTE THAT ONLY VALUES WHICH ARE THE RESULT OF A
C           COMPUTATION MAY BE OUTPUT.  CONSTANTS OR DATA VALUES
C           MAY NOT BE OUTPUT.
C
C  4. THE REAL ASSIGNMENT STATEMENT
C
C     USE:  TO ASSIGN A VALUE TO A REAL VARIABLE
C
C     FORM: REAL VARIABLE = REAL EXPRESSION
C
C           WHERE THE REAL VARIABLE IS EITHER A REAL SCALAR OR SINGLE
C           ARRAY ELEMENT.  REAL EXPRESSIONS ARE MADE UP OF REAL
C           VARIABLES AND CONSTANTS COMBINED WITH THE BINARY OPERATORS
C           +, -, * AND / AND THE UNARY OPERATORS UNARY - AND SQRT.
C           (THE OPERAND OF THE SQRT MUST APPEAR IN PARENTHESES.)
C           (NOTE THAT THERE IS NO REAL EXPONENTIATION ALLOWED.)
C           OPERATOR PRECEDENCE IS AS IN STANDARD FORTRAN. IN ADDITION
C           THERE IS A SUMMATION OPERATION ON ARRAY VECTORS, IN EFFECT,
C           A BUILT-IN INNER PRODUCT.  A SUMMATION EXPRESSION CAN APPEAR
C           IN A REAL EXPRESSION AND IS OF THE FORM:
C
C           SUMMATION(D1 * D2, SUMMATION-VARIABLE = INTEXP1 TO INTEXP2)
C
C           WHERE INTEXP IS AN INTEGER EXPRESSION, THE SUMMATION
C           VARIABLE IS ANY INTEGER IDENTIFIER NAME, AND WHERE D1 AND D2
C           ARE EACH OF ONE OF THE FORMS:
C               ARRAY NAME(SUMMATION-VARIABLE)
C               ARRAY NAME(SUMMATION-VARIABLE,SUMMATION-VARIABLE)
C               ARRAY NAME(SUMMATION-VARIABLE,INTEXP)
C               ARRAY NAME(INTEXP,SUMMATION-VARIABLE)
C
C           A SUMMATION EXPRESSION WILL BE INTERPRETED AS AN IMPLICIT
C           FOR-LOOP. ANY USE OF THE SUMMATION VARIABLE IN THE INTEGER
C           EXPRESSIONS BOUNDING THE SUMMATION LOOP WILL BE FLAGGED AS
C           AN ERROR.
C
C  5.  THE FOR STATEMENT
C
C     USE:  AS A MEANS OF INDICATING THAT A BLOCK OF STATEMENTS IS TO BE
C           ITERATIVELY EXECUTED A SPECIFIED NUMBER OF TIMES.
C
C     FORM:  FOR INTEGER-VARIABLE = INTEXP1 TO INTEXP2 BY INCREMENT
C
C           WHERE INTEXP STANDS FOR INTEGER EXPRESSION, AND INCREMENT IS
C           WRITTEN AS EITHER +1, -1 OR 1.
C
C     INTERPRETATION:  ALL STATEMENTS UP TO THE END STATEMENT MATCHING
C           THIS FOR STATEMENT (SEE BELOW FOR DISCUSSION OF END
C           STATEMENTS) WILL BE ITERATIVELY INTERPRETED AS IN A FORTRAN
C           DO LOOP, EXCEPT THAT NEGATIVE INCREMENTS ARE ALLOWED AND IN
C           THIS CASE, THE LOOP VARIABLE TEST IS DONE AT THE TOP OF THE
C           LOOP.  THUS, EMPTY LOOPS ARE POSSIBLE, THAT IS THOSE WHICH
C           WILL NOT BE EXECUTED AT ALL.  (NOTE THE SAME IS TRUE OF THE
C           IMPLICIT FOR-LOOP IN A SUMMATION EXPRESSION.)
C
C  6. THE IF-THEN STATEMENT
C
C     USE:  TO ALLOW SELECTIVE EXECUTION OF A BLOCK OF STATEMENTS
```

```
C           DEPENDING ON THE OUTCOME OF A COMPARISON OF THE VALUES OF
C           TWO INTEGER EXPRESSIONS.
C
C     FORM: IF INTEXP1 .R. INTEXP2 THEN
C
C           WHERE INTEXP STANDS FOR INTEGER EXPRESSION AND R IS ONE OF
C           THE RELATIONS WRITTEN EQ,NE,GT,LT,LE OR GE WITH THE
C           STANDARD FORTRAN DENOTATION. NOTE THAT THERE ARE NO
C           PARENTHESES AROUND THE RELATIONAL EXPRESSION.
C
C     INTERPRETATION: IF THE TEST SUCCEEDS, THAT IS IF THE TWO INTEGER
C           EXPRESSIONS ARE RELATED IN THE INDICATED WAY AT THE TIME OF
C           INTERPRETATION, THEN ALL THE STATEMENTS UP TO THE NEXT END
C           STATEMENT WILL BE INTERPRETED.  OTHERWISE, THE FIRST
C           EXECUTABLE STATEMENT FOLLOWING THE NEXT END STATEMENT WILL
C           BE THE NEXT STATEMENT INTERPRETED.
C
C  7. THE END STATEMENT
C
C     USE:  TO DEFINE THE ENDS OF BLOCKS OF STATEMENTS BEGINNING WITH
C           FOR STATEMENTS OR IF-THEN STATEMENTS.
C
C     FORM1:    END
C     FORM2 :   END(INTEGER-VARIABLE)
C
C     MEANING:  WHEN FORM 1 IS USED THE EFFECT IS TO CLOSE THE BLOCK OF
C           STATEMENTS  BEGINNING AT THE NEAREST PRECEEDING FOR OR
C           IF-THEN STATEMENT.
C           WHEN FORM2 IS USED THE EFFECT IS TO CLOSE THE FOR BLOCK
C           WHOSE LOOP VARIABLE MATCHES THE END STATEMENT VARIABLE.  IN
C           ADDITION, ANY FOR OR IF-THEN BLOCKS WHICH BEGIN BETWEEN THIS
C           END STATEMENT AND ITS MATCHING FOR STATEMENT ARE CLOSED.
C           THIS INTERPRETATION IMPOSES STANDARD FORTRAN LIKE NESTING
C           CONVENTIONS ON FOR AND IF-THEN BLOCKS.  THAT IS, A SEQUENCE
C           OF STATEMENTS
C                           FOR K = IE1 TO IE2 BY 1
C                           .
C                           .
C                           FOR I = IE3 TO IE4 BY 1
C                           .
C                           .
C                           END(K)
C                           .
C                           END(I)
C           WILL RESULT IN AN ERROR MESSAGE WHEN THE END(I) STATEMENT
C           IS ENCOUNTERED, BECAUSE BOTH FOR STATEMENTS WILL HAVE BEEN
C           CLOSED BY THE PARSER WHEN THE END(K) STATEMENT WAS PARSED.
C
C  NOTE:    ADDITIONAL RESTRICTIONS ON BLOCK STRUCTURES:
C           1) AT MOST EIGHT FOR AND/OR IF-THEN BLOCKS CAN BE BEGUN
C              BEFORE AN END STATEMENT OCCURS.
C           2) A FOR LOOP VARIABLE CANNOT BE USED AGAIN AS AN EXPLICIT
C              FOR LOOP VARIABLE WITHIN ITS ORIGINAL LOOP. IT CAN BE
C              REUSED AS A SUMMATION VARIABLE, HOWEVER.
C
C  8. THE STOP STATEMENT
C
C     USE:  TO DENOTE THE END OF THE PROGRAM
C
C     FORM:  *STOP
C
```

Contents of program comparing rounding error in a single algorithm with perturbations of the problem: MAIN1(372), F(64), ROUND(116), GETRHO(64), OMEGA(76)

```
C
C THE USER SUPPLIES:
C
C     THE OUTPUT FROM THE MINICOMPILER.
C
C     THE ENTRIES OF THE INITIAL SET OF DATA.  ONE ENTRY PER CARD, EACH
C     ENTRY WRITTEN WITH A DECIMAL POINT AND CONTAINED IN THE FIRST
C     TWENTY COLUMNS.
C
C     THE CHOICE OF OMEGA (+) OR RHO (-).  FORMAT(I2).
```

```
C       CODE : PERTURBATIONS                                                    ØØØØØ55Ø
C         Ø1 : IN THE DATA, MEASURED COMPONENT-WISE.                            ØØØØØ6ØØ
C         Ø2 : IN THE DATA AND RESULT, MEASURED COMPONENT-WISE.                 ØØØØØ65Ø
C         Ø3 : IN THE DATA, MEASURED NORM-WISE.                                 ØØØØØ7ØØ
C         Ø4 : IN THE DATA AND RESULT, MEASURED NORM-WISE.                      ØØØØØ75Ø
C                                                                               ØØØØØ8ØØ
C     THE STOPPING VALUE FOR THE MAXIMIZER.  WRITTEN WITH A DECIMAL             ØØØØØ85Ø
C     POINT AND CONTAINED IN THE FIRST TWENTY COLUMNS OF ITS CARD.              ØØØØØ9ØØ
C                                                                               ØØØØØ95Ø
C THE SOFTWARE RETURNS:                                                         ØØØØ1ØØØ
C                                                                               ØØØØ1Ø5Ø
C     AN ANNOTATED LISTING OF THE USER-SUPPLIED INFORMATION, PLUS THE           ØØØØ11ØØ
C     ERROR-COMPARING VALUE, THE CONSTRAINT VALUES (IF ANY) AND                 ØØØØ115Ø
C     THE OUTPUT COMPUTED AT THE INITIAL SET OF DATA.                           ØØØØ12ØØ
C                                                                               ØØØØ125Ø
C     A LIST OF SELECTED VALUES FOUND BY THE MAXIMIZER.                         ØØØØ13ØØ
C                                                                               ØØØØ135Ø
C     THE FINAL SET OF DATA.                                                    ØØØØ14ØØ
C                                                                               ØØØØ145Ø
C     IF INSTABILITY IS DIAGNOSED, THEN ALL ARITHMETIC OPERATIONS               ØØØØ15ØØ
C     AT THE FINAL SET OF DATA ARE LISTED.                                      ØØØØ155Ø
C                                                                               ØØØØ16ØØ
C OTHER INFORMATION IS RETURNED IF EXCEPTIONS ARISE.                            ØØØØ165Ø
C                                                                               ØØØØ17ØØ
C     ------------------------------------------------------------------------ØØØØ175Ø
C                                                                               ØØØØ18ØØ
C THE USER CAN AVOID THE MINICOMPILER BY SUPPLYING:                             ØØØØ185Ø
C                                                                               ØØØØ19ØØ
C     THE NUMBER OF OPERATIONS IN THE PROGRAM BEING TESTED.  GE.1               ØØØØ195Ø
C     AND LE.2ØØ.  FORMAT(I3).  (INSTRUCTIONS ARE PROVIDED BELOW FOR            ØØØØ2ØØØ
C     RAISING THE UPPER BOUND TO TEST LONGER PROGRAMS, OR LOWERING IT           ØØØØ2Ø5Ø
C     TO CONSERVE STORAGE.)                                                     ØØØØ21ØØ
C                                                                               ØØØØ215Ø
C     THE OPERATIONS OF THE STRAIGHT-LINE PROGRAM.  FORMAT(I3,I2,I4).           ØØØØ22ØØ
C     THE I-TH DATA ENTRY IS ENCODED AS I, THE J-TH COMPUTED VALUE AS           ØØØØ225Ø
C     1ØØ + J AND THE K-TH CONSTANT AS -K.  THE OPERATIONS +, -, *, /,          ØØØØ23ØØ
C     SQRT AND UNARY MINUS ARE ENCODED AS 1-6, RESPECTIVELY.  SQRT AND          ØØØØ235Ø
C     UNARY MINUS REQUIRE Ø AS THEIR SECOND OPERAND.                            ØØØØ24ØØ
C                                                                               ØØØØ245Ø
C     THE NUMBER OF OUTPUTS OF THE STRAIGHT-LINE PROGRAM.                       ØØØØ25ØØ
C     GE.1 AND LE.2Ø.  FORMAT(I2)                                               ØØØØ255Ø
C                                                                               ØØØØ26ØØ
C     THE INSTRUCTIONS AT WHICH THE OUTPUTS ARE COMPUTED.  FORMAT(I3).          ØØØØ265Ø
C                                                                               ØØØØ27ØØ
C     THE NUMBER OF CONSTANTS.  LE.2Ø.  FORMAT(I2).                             ØØØØ275Ø
C                                                                               ØØØØ28ØØ
C     THE CONSTANTS.  FORMAT(G2Ø.16)                                            ØØØØ285Ø
C                                                                               ØØØØ29ØØ
C     THE NUMBER OF ENTRIES IN A SET OF DATA.  LE.3Ø.  FORMAT(I2).              ØØØØ295Ø
C                                                                               ØØØØ3ØØØ
C      -----------------------------------------------------------------------ØØØØ3Ø5Ø
C                                                                               ØØØØ31ØØ
C THIS MAIN PROGRAM PERFORMS INPUT AND OUTPUT DUTIES.                           ØØØØ315Ø
C                                                                               ØØØØ32ØØ
C THE SUBPROGRAMS ARE:                                                          ØØØØ325Ø
C                                                                               ØØØØ33ØØ
C     MAXIM - A 'DIRECT SEARCH' NUMERICAL MAXIMIZER CALLED BY THE               ØØØØ335Ø
C     MAIN PROGRAM.                                                             ØØØØ34ØØ
C                                                                               ØØØØ345Ø
C     GRAM - A GRAM-SCHMIDT ROUTINE USED BY MAXIM.                              ØØØØ35ØØ
C                                                                               ØØØØ355Ø
C     F - FUNCTION CALLED BY MAXIM WHICH EVALUATES THE PENALIZED                ØØØØ36ØØ
C     ERROR-COMPARING VALUE.                                                    ØØØØ365Ø
C                                                                               ØØØØ37ØØ
C     ROUND - ROUTINE CALLED BY THE MAIN PROGRAM AND F TO EVALUATE              ØØØØ375Ø
C     SENSITIVITY TO ERRORS.                                                    ØØØØ38ØØ
C                                                                               ØØØØ385Ø
C     GETRHO - ROUTINE CALLED BY ROUND IF THE USER OPTS TO TEST RHO             ØØØØ39ØØ
C     INSTEAD OF OMEGA.                                                         ØØØØ395Ø
C                                                                               ØØØØ4ØØØ
C     OMEGA - ROUTINE CALLED BY ROUND TO EVALUATE OMEGA.                        ØØØØ4Ø5Ø
C                                                                               ØØØØ41ØØ
C     GENEIG, REDUC, TRED1, TRIDIB - ROUTINES TO COMPUTE THE                    ØØØØ415Ø
C     EIGENVALUES FOR OMEGA.  THE USER MAY NEED TO USE DIFFERENT                ØØØØ42ØØ
C     VERSIONS OF THE EISPACK ROUTINES.                                         ØØØØ425Ø
C                                                                               ØØØØ43ØØ
C     POSITV - USER-SUPPLIED ROUTINE TO EVALUATE CONSTRAINTS.                   ØØØØ435Ø
C                                                                               ØØØØ44ØØ
```

Contents of program comparing rounding errors in two algorithms: MAIN2(354), F(64), ROUND(111), GETRHO(36), OMEGA(40)

```
C                                                                       ØØØØØØ5Ø
C THE USER SUPPLIES:                                                    ØØØØØ1ØØ
C                                                                       ØØØØØ15Ø
C     THE OUTPUT FROM THE MINICOMPILER FOR EACH OF THE TWO              ØØØØØ2ØØ
C     ALGORITHMS BEING TESTED.                                          ØØØØØ25Ø
C                                                                       ØØØØØ3ØØ
C     THE ENTRIES OF THE INITIAL SET OF DATA.  ONE ENTRY PER CARD, EACH ØØØØØ35Ø
C     ENTRY WRITTEN WITH A DECIMAL POINT AND CONTAINED IN THE FIRST     ØØØØØ4ØØ
C     TWENTY COLUMNS.                                                   ØØØØØ45Ø
C                                                                       ØØØØØ5ØØ
C     THE CHOICE OF OMEGA (+) OR RHO (-).  FORMAT(I2).                  ØØØØØ55Ø
C       CODE : COMPARE                                                  ØØØØØ6ØØ
C         Ø1 : (ALGORITHM 1) / (ALGORITHM 2)                            ØØØØØ65Ø
C         Ø2 : (ALGORITHM 2) / (ALGORITHM 1)                            ØØØØØ7ØØ
C                                                                       ØØØØØ75Ø
C     THE STOPPING VALUE FOR THE MAXIMIZER.  WRITTEN WITH A DECIMAL     ØØØØØ8ØØ
C     POINT AND CONTAINED IN THE FIRST TWENTY COLUMNS OF ITS CARD.      ØØØØØ85Ø
C                                                                       ØØØØØ9ØØ
C THE SOFTWARE RETURNS:                                                 ØØØØØ95Ø
C                                                                       ØØØØ1ØØØ
C     AN ANNOTATED LISTING OF THE USER-SUPPLIED INFORMATION, PLUS THE   ØØØØ1Ø5Ø
C     ERROR-COMPARING VALUE, THE CONSTRAINT VALUES (IF ANY) AND         ØØØØ11ØØ
C     THE OUTPUT COMPUTED AT THE INITIAL SET OF DATA.                   ØØØØ115Ø
C                                                                       ØØØØ12ØØ
C     A LIST OF SELECTED VALUES FOUND BY THE MAXIMIZER.                 ØØØØ125Ø
C                                                                       ØØØØ13ØØ
C     THE FINAL SET OF DATA.                                            ØØØØ135Ø
C                                                                       ØØØØ14ØØ
C OTHER INFORMATION IS RETURNED IF EXCEPTIONS ARISE.                    ØØØØ145Ø
```

Contents required by both the above rounding error packages and the data for the minicompiler: LISTOP(38), MAXIM(109), GRAM(38), GENEIG(44), REDUC(119), TRED1(118), TRIDIB(272), MDATA^{D}(279)

```
C                                                                       ØØØØ15ØØ
C     ------------------------------------------------------------------ØØØØ155Ø
C                                                                       ØØØØ16ØØ
C THE USER CAN AVOID THE MINICOMPILER BY SUPPLYING:                     ØØØØ165Ø
C                                                                       ØØØØ17ØØ
C     THE NUMBER OF OPERATIONS IN THE PROGRAM BEING TESTED.  GE.1       ØØØØ175Ø
C     AND LE.2ØØ.  FORMAT(I3).  (INSTRUCTIONS ARE PROVIDED BELOW FOR    ØØØØ18ØØ
C     RAISING THE UPPER BOUND TO TEST LONGER PROGRAMS, OR LOWERING IT   ØØØØ185Ø
C     TO CONSERVE STORAGE.)                                             ØØØØ19ØØ
C                                                                       ØØØØ195Ø
C     THE OPERATIONS OF THE STRAIGHT-LINE PROGRAM.  FORMAT(I3,I2,I4).   ØØØØ2ØØØ
C     THE I-TH DATA ENTRY IS ENCODED AS I, THE J-TH COMPUTED VALUE AS   ØØØØ2Ø5Ø
C     1ØØ + J AND THE K-TH CONSTANT AS -K.  THE OPERATIONS +, -, *, /,  ØØØØ21ØØ
C     SQRT AND UNARY MINUS ARE ENCODED AS 1-6, RESPECTIVELY.  SQRT AND  ØØØØ215Ø
C     UNARY MINUS REQUIRE Ø AS THEIR SECOND OPERAND.                    ØØØØ22ØØ
C                                                                       ØØØØ225Ø
C     THE NUMBER OF OUTPUTS OF THE STRAIGHT-LINE PROGRAM.               ØØØØ23ØØ
C     GE.1 AND LE.2Ø.  FORMAT(I2)                                       ØØØØ235Ø
C                                                                       ØØØØ24ØØ
C     THE INSTRUCTIONS AT WHICH THE OUTPUTS ARE COMPUTED.  FORMAT(I3).  ØØØØ245Ø
C                                                                       ØØØØ25ØØ
C     THE NUMBER OF CONSTANTS.  LE.2Ø.  FORMAT(I2).                     ØØØØ255Ø
C                                                                       ØØØØ26ØØ
C     THE CONSTANTS.  FORMAT(G2Ø.16)                                    ØØØØ265Ø
C                                                                       ØØØØ27ØØ
C     THE NUMBER OF ENTRIES IN A SET OF DATA.  LE.3Ø.  FORMAT(I2).      ØØØØ275Ø
C                                                                       ØØØØ28ØØ
C      ------------------------------------------------------------------ØØØØ285Ø
C                                                                       ØØØØ29ØØ
C THIS MAIN PROGRAM PERFORMS INPUT AND OUTPUT DUTIES.                   ØØØØ295Ø
C                                                                       ØØØØ3ØØØ
C THE SUBPROGRAMS ARE:                                                  ØØØØ3Ø5Ø
C                                                                       ØØØØ31ØØ
C     MAXIM - A 'DIRECT SEARCH' NUMERICAL MAXIMIZER CALLED BY THE       ØØØØ315Ø
C     MAIN PROGRAM.                                                     ØØØØ32ØØ
C                                                                       ØØØØ325Ø
C     GRAM - A GRAM-SCHMIDT ROUTINE USED BY MAXIM.                      ØØØØ33ØØ
C                                                                       ØØØØ335Ø
C     F - FUNCTION CALLED BY MAXIM WHICH EVALUATES THE PENALIZED        ØØØØ34ØØ
```

```
C     ERROR-COMPARING VALUE.                                          0000345Ø
C                                                                     0000350Ø
C     ROUND - ROUTINE CALLED BY THE MAIN PROGRAM AND F TO EVALUATE    0000355Ø
C     SENSITIVITY TO ERRORS.                                          0000360Ø
C                                                                     0000365Ø
C     GETRHO - ROUTINE CALLED BY ROUND IF THE USER OPTS TO TEST RHO   0000370Ø
C     INSTEAD OF OMEGA.                                               0000375Ø
C                                                                     0000380Ø
C     OMEGA - ROUTINE CALLED BY ROUND TO EVALUATE OMEGA.              0000385Ø
C                                                                     0000390Ø
C     GENEIG, REDUC, TRED1, TRIDIB - ROUTINES TO COMPUTE THE          0000395Ø
C     EIGENVALUES FOR OMEGA.  THE USER MAY NEED TO USE DIFFERENT      0000400Ø
C     VERSIONS OF THE EISPACK ROUTINES.                               0000405Ø
C                                                                     0000410Ø
C     POSITV - USER-SUPPLIED ROUTINE TO EVALUATE CONSTRAINTS.         0000415Ø
C                                                                     0000420Ø
```

ALGORITHM 533
NSPIV, A Fortran Subroutine for Sparse Gaussian Elimination With Partial Pivoting [F4]

ANDREW H. SHERMAN
The University of Texas at Austin

Key Words and Phrases: sparse Gaussian elimination, sparse linear systems, linear equations, partial pivoting algorithms
CR Categories: 5.14
Language: Fortran

DESCRIPTION

1. Introduction

NSPIV is a Fortran subroutine which solves a sparse system of linear equations

$$Ax = b$$

by sparse Gaussian elimination with partial pivoting. More precisely, it performs Gaussian elimination with column interchanges on the nonsingular $N \times N$ matrix A to effectively obtain a factorization of the form

$$AQ = LU,$$

where L is lower triangular, U is unit upper triangular, and Q is a permutation matrix corresponding to the column interchanges. To conserve storage, only the factor U is retained, so during elimination, operations are performed on the right-hand side to obtain the solution y of the system.

$$Ly = b.$$

Once U has been obtained, x is computed by solving the upper triangular system

$$UQ^{\mathrm{T}}x = y.$$

This algorithm discusses the usage of NSPIV and gives a few test results. The method used in NSPIV is described in [7], where it is called "run insertion"; [7] also includes extensive test results which show NSPIV to be more efficient than other currently available software for sparse Gaussian elimination with pivoting.

Received 17 June 1977 and 8 November 1977.

This work was supported in part by the National Science Foundation under Grant NSF DCR 73-07998, in part by the Energy Research and Development Administration under Grant US ERDA E(11-1) 2383, and in part by Yale University under a grant subcontracted from the United States Air Force Office of Scientific Research.
Author's address: Department of Computer Sciences, Painter 328, The University of Texas at Austin, Austin, TX 78712.

2. Usage

For a description of the calling sequence and matrix storage schemes used, see the listing presented here.

The actual numerical computations are performed in an internal subroutine NSPIV1, which is written to perform all computations in single precision. Conversion to double precision may be accomplished simply by changing REAL declarations to DOUBLE PRECISION declarations in both NSPIV and NSPIV1, and by changing the calls to ABS into calls to DABS.

It is well known that the initial ordering of the rows of A greatly affects the overall performance of a subroutine like NSPIV (cf. [1, 7]). To provide initial row and column orderings to NSPIV, the user must set the arrays R, C, and IC so that $R(I)$ is the number of the Ith row in the row ordering, $C(I)$ is the number of the Ith column in the column ordering, and $IC(C(I)) = I$ for all I. (Notice that no modifications are required to the arrays, IA, JA, and A.) It is always assumed by NSPIV that the user has provided initial row and column orderings; if the orderings are unchanged from those of A, then the user should set $R(I) = C(I) = IC(I) = I$.

Often, sparse linear systems arise with a natural band structure in which all of the nonzeros of A are clustered near the main diagonal. On such problems, with a poor initial ordering, NSPIV may perform worse than some widely available subroutines for band Gaussian elimination with partial pivoting. However, for a sparse matrix A with a given row ordering, NSPIV will often require less storage and not too much more computation time than a band subroutine. Moreover, it is usually possible to find easily computed row orderings for which NSPIV is substantially more efficient than a band subroutine would be with a good band-reducing ordering.

To illustrate these remarks, we consider a sample problem which could arise in the numerical solution of partial differential equations. Here A is a 10×10 block tridiagonal matrix

$$A = \begin{pmatrix} C & D & & & \\ B & C & D & & \\ & B & \ddots & \ddots & \\ & & \ddots & \ddots & D \\ & & & B & C \end{pmatrix}$$

with the 10×10 blocks given by

$$C = \begin{pmatrix} 3. & -.5 & & & \\ -1.5 & 3. & -.5 & & \\ & -1.5 & \ddots & \ddots & \\ & & \ddots & \ddots & -.5 \\ & & & -1.5 & 3. \end{pmatrix}, \quad B = D = \begin{pmatrix} -1. & & & \\ & -1. & & \\ & & \ddots & \\ & & & -1. \end{pmatrix}$$

We solved this problem using NSPIV with three different initial row orderings: the natural ordering (NO) in which the problem was presented, the so-called alternate diagonal ordering (ADO) (cf. Price and Coats [6]), and an ordering in which the sparsest rows are ordered first (SO). We also solved the problem using the subroutine LEQT1B from the IMSL library ([4]) using the natural ordering to produce the smallest possible bandwidth for this problem (cf. George [3]).

Table I

Subroutine	Ordering	Solution Time*	Total Storage
NSPIV	SO	.799	4785
NSPIV	ADO	.146	2453
NSPIV	NO	.283	3181
LEQTIB	NO	.244	3300

* Times are in seconds. Programs were run on a CDC-6600 computer using the MNF Fortran compiler.

Our results are summarized in Table I. With the SO ordering, NSPIV required much more storage and time than LEQT1B did with the natural ordering. With identical orderings, LEQT1B was slightly faster than NSPIV, but NSPIV required less storage and also saved both the input matrix A and the right-hand side b in their original forms. (LEQT1B does save both triangular factors, however.) Finally, with the ADO ordering, NSPIV used substantially less storage and computation time than LEQT1B. It should be noted in passing that the SO ordering apparently works quite well on problems without the natural band structure present in this example (cf. [1, 7]), even though it did not do well here. Also, experiments indicate that more complex orderings such as the minimum degree or modified Markowitz orderings (cf. Sherman [8]) may be used to good advantage with NSPIV.

REFERENCES

1. CURTIS, A.R., AND REID, J.K. The solution of large sparse unsymmetric systems of linear equations. *Information Processing '71* (1971), 1240–1245.
2. FORSYTHE, G.E., AND MOLER, C.B. *Computer Solution of Linear Algebraic Equations.* Prentice-Hall, Englewood Cliffs, N.J., 1967.
3. GEORGE, J.A. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal. 10* (1973), 345–363.
4. International Mathematical and Statistical Libraries, Inc. The IMSL Library 3, Edition 6, 1977.
5. KNUTH, D.E. *The Art of Computer Programming, Vol. 3: Searching and Sorting.* Addison-Wesley, Reading, Mass., 1973.
6. PRICE, H.S., AND COATS, K.H. Direct methods in reservoir simulation. Paper SPE 4278, presented at the Meeting of the Soc. of Petroleum Engineers of the AIME, 1973.
7. SHERMAN, A.H. Algorithms for sparse Gaussian elimination with partial pivoting. *ACM Trans. Math. Software 4*, 4 (Dec. 1978), 330–338.
8. SHERMAN, A.H. On the efficient solution of sparse systems of linear and nonlinear equations. Ph.D. Diss., Dep. Computer Science, Yale U., 1973.

ALGORITHM

NAME(n): indicates a Fortran module with n records
NAME^T: indicates "NAME" is included for testing purposes
NAME^D: indicates "NAME" contains test data
Contents: NSPIV(108), NSPIV1(275), MAIN^T(62), PREORD^T(35), RESCHK^T(28), PIVCHK^T(35), GENPRB^T(37), DATA^D(960)

```
      SUBROUTINE NSPIV (N,IA,JA,A,B,MAX,R,C,IC,X,ITEMP,RTEMP,IERR)          1
C                                                                           2
C                                                                           3
C  NSPIV CALLS NSPIV1 WHICH USES SPARSE GAUSSIAN ELIMINATION WITH           4
C  COLUMN INTERCHANGES TO SOLVE THE LINEAR SYSTEM A X = B.  THE             5
C  ELIMINATION PHASE PERFORMS ROW OPERATIONS ON A AND B TO OBTAIN           6
C  A UNIT UPPER TRIANGULAR MATRIX U AND A VECTOR Y.  THE SOLUTION           7
C  PHASE SOLVES U X = Y.                                                    8
C                                                                           9
C                                                                           1Ø
C  INPUT ARGUMENTS---                                                       11
C                                                                           12
C  N       INTEGER NUMBER OF EQUATIONS AND UNKNOWNS                         13
C                                                                           14
C  IA      INTEGER ARRAY OF N+1 ENTRIES CONTAINING ROW POINTERS TO A        15
C          (SEE MATRIX STORAGE DESCRIPTION BELOW)                           16
C                                                                           17
```

```
C  JA      INTEGER ARRAY WITH ONE ENTRY PER NONZERO IN A, CONTAINING          18
C          COLUMN NUMBERS OF THE NONZEROES OF A.  (SEE MATRIX STORAGE        19
C          DESCRIPTION BELOW)                                                2Ø
C                                                                            21
C  A       REAL ARRAY WITH ONE ENTRY PER NONZERO IN A, CONTAINING THE         22
C          ACTUAL NONZEROES.  (SEE MATRIX STORAGE DESCRIPTION BELOW)         23
C                                                                            24
C  B       REAL ARRAY OF N ENTRIES CONTAINING RIGHT HAND SIDE DATA            25
C                                                                            26
C  MAX     INTEGER NUMBER SPECIFYING MAXIMUM NUMBER OF OFF-DIAGONAL           27
C          NONZERO ENTRIES OF U WHICH MAY BE STORED                          28
C                                                                            29
C  R       INTEGER ARRAY OF N ENTRIES SPECIFYING THE ORDER OF THE             3Ø
C          ROWS OF A (I.E., THE ELIMINATION ORDER FOR THE EQUATIONS)         31
C                                                                            32
C  C       INTEGER ARRAY OF N ENTRIES SPECIFYING THE ORDER OF THE             33
C          COLUMNS OF A.  C IS ALSO AN OUTPUT ARGUMENT                       34
C                                                                            35
C  IC      INTEGER ARRAY OF N ENTRIES WHICH IS THE INVERSE OF C               36
C          (I.E., IC(C(I)) = I).  IC IS ALSO AN OUTPUT ARGUMENT              37
C                                                                            38
C  ITEMP   INTEGER ARRAY OF 2*N + MAX + 2 ENTRIES, FOR INTERNAL USE          39
C                                                                            4Ø
C  RTEMP   REAL ARRAY OF N + MAX ENTRIES FOR INTERNAL USE                    41
C                                                                            42
C                                                                            43
C  OUTPUT ARGUMENTS---                                                       44
C                                                                            45
C  C       INTEGER ARRAY OF N ENTRIES SPECIFYING THE ORDER OF THE             46
C          COLUMNS OF U.  C IS ALSO AN INPUT ARGUMENT                        47
C                                                                            48
C  IC      INTEGER ARRAY OF N ENTRIES WHICH IS THE INVERSE OF C               49
C          (I.E., IC(C(I)) = I).  IC IS ALSO AN INPUT ARGUMENT               5Ø
C                                                                            51
C  X       REAL ARRAY OF N ENTRIES CONTAINING THE SOLUTION VECTOR             52
C                                                                            53
C  IERR    INTEGER NUMBER WHICH INDICATES ERROR CONDITIONS OR                 54
C          THE ACTUAL NUMBER OF OFF-DIAGONAL ENTRIES IN U (FOR               55
C          SUCCESSFUL COMPLETION)                                            56
C                                                                            57
C          IERR VALUES ARE---                                                58
C                                                                            59
C          Ø LT IERR              SUCCESSFUL COMPLETION.  U HAS IERR         6Ø
C                                 OFF-DIAGONAL NONZERO ENTRIES               61
C                                                                            62
C          IERR = Ø               ERROR.  N = Ø                              63
C                                                                            64
C          -N LE IERR LT Ø        ERROR.  ROW NUMBER IABS(IERR) OF A IS      65
C                                 IS NULL                                    66
C                                                                            67
C          -2*N LE IERR LT -N     ERROR.  ROW NUMBER IABS(IERR+N) HAS A      68
C                                 DUPLICATE ENTRY                            69
C                                                                            7Ø
C          -3*N LE IERR LT -2*N   ERROR.  ROW NUMBER IABS(IERR+2*N)          71
C                                 HAS A ZERO PIVOT                           72
C                                                                            73
C          -4*N LE IERR LT -3*N   ERROR.  ROW NUMBER IABS(IERR+3*N)          74
C                                 EXCEEDS STORAGE                            75
C                                                                            76
C                                                                            77
C  STORAGE OF SPARSE MATRICES---                                             78
C                                                                            79
C  THE SPARSE MATRIX A IS STORED USING THREE ARRAYS IA, JA, AND A.           8Ø
C  THE ARRAY A CONTAINS THE NONZEROES OF THE MATRIX ROW-BY-ROW, NOT          81
C  NECESSARILY IN ORDER OF INCREASING COLUMN NUMBER.  THE ARRAY JA           82
C  CONTAINS THE COLUMN NUMBERS CORRESPONDING TO THE NONZEROES STORED         83
C  IN THE ARRAY A (I.E., IF THE NONZERO STORED IN A(K) IS IN                 84
C  COLUMN J, THEN JA(K) = J).  THE ARRAY IA CONTAINS POINTERS TO THE         85
C  ROWS OF NONZEROES/COLUMN INDICES IN THE ARRAY A/JA (I.E.,                 86
C  A(IA(I))/JA(IA(I)) IS THE FIRST ENTRY FOR ROW I IN THE ARRAY A/JA).       87
C  IA(N+1) IS SET SO THAT IA(N+1) - IA(1) = THE NUMBER OF NONZEROES IN A     88
C                                                                            89
C                                                                            9Ø
      REAL A(1),B(1),X(1),RTEMP(1)                                           91
      INTEGER IA(1),JA(1),R(1),C(1),IC(1),ITEMP(1)                           92
      INTEGER IU,JU,U,Y,P                                                    93
```

```
C                                                                       94
C  SET INDICES TO DIVIDE TEMPORARY STORAGE FOR NSPIV1                   95
C                                                                       96
      Y = 1                                                             97
      U = Y + N                                                         98
      P = 1                                                             99
      IU = P + N + 1                                                   1ØØ
      JU = IU + N + 1                                                  1Ø1
C                                                                      1Ø2
C  CALL NSPIV1 TO PERFORM COMPUTATIONS                                 1Ø3
C                                                                      1Ø4
      CALL NSPIV1 (N,IA,JA,A,B,MAX,R,C,IC,X,RTEMP(Y),ITEMP(P),         1Ø5
     C             ITEMP(IU),ITEMP(JU),RTEMP(U),IERR)                  1Ø6
      RETURN                                                           1Ø7
      END                                                              1Ø8
      SUBROUTINE NSPIV1 (N,IA,JA,A,B,MAX,R,C,IC,X,Y,P,IU,JU,U,IERR)    1Ø9
C                                                                      11Ø
C                                                                      111
C  NSPIV1 USES SPARSE GAUSSIAN ELIMINATION WITH                        112
C  COLUMN INTERCHANGES TO SOLVE THE LINEAR SYSTEM A X = B.  THE        113
C  ELIMINATION PHASE PERFORMS ROW OPERATIONS ON A AND B TO OBTAIN      114
C  A UNIT UPPER TRIANGULAR MATRIX U AND A VECTOR Y.  THE SOLUTION      115
C  PHASE SOLVES U X = Y.                                               116
C                                                                      117
C                                                                      118
C  SEE NSPIV FOR DESCRIPTIONS OF ALL INPUT AND OUTPUT ARGUMENTS        119
C  OTHER THAN THOSE DESCRIBED BELOW                                    12Ø
C                                                                      121
C  INPUT ARGUMENTS (USED INTERNALLY ONLY)---                           122
C                                                                      123
C  Y   REAL ARRAY OF N ENTRIES USED TO COMPUTE THE UPDATED             124
C      RIGHT HAND SIDE                                                 125
C                                                                      126
C  P   INTEGER ARRAY OF N+1 ENTRIES USED FOR A LINKED LIST.            127
C      P(N+1) IS THE LIST HEADER, AND THE ENTRY FOLLOWING              128
C      P(K) IS IN P(P(K)).  THUS, P(N+1) IS THE FIRST DATA             129
C      ITEM, P(P(N+1)) IS THE SECOND, ETC.  A POINTER OF               13Ø
C      N+1 MARKS THE END OF THE LIST                                   131
C                                                                      132
C  IU  INTEGER ARRAY OF N+1 ENTRIES USED FOR ROW POINTERS TO U         133
C      (SEE MATRIX STORAGE DESCRIPTION BELOW)                          134
C                                                                      135
C  JU  INTEGER ARRAY OF MAX ENTRIES USED FOR COLUMN NUMBERS OF         136
C      THE NONZEROES IN THE STRICT UPPER TRIANGLE OF U.  (SEE          137
C      MATRIX STORAGE DESCRIPTION BELOW)                               138
C                                                                      139
C  U   REAL ARRAY OF MAX ENTRIES USED FOR THE ACTUAL NONZEROES IN      14Ø
C      THE STRICT UPPER TRIANGLE OF U.  (SEE MATRIX STORAGE            141
C      DESCRIPTION BELOW)                                              142
C                                                                      143
C                                                                      144
C  STORAGE OF SPARSE MATRICES---                                       145
C                                                                      146
C  THE SPARSE MATRIX A IS STORED USING THREE ARRAYS IA, JA, AND A.     147
C  THE ARRAY A CONTAINS THE NONZEROES OF THE MATRIX ROW-BY-ROW, NOT    148
C  NECESSARILY IN ORDER OF INCREASING COLUMN NUMBER.  THE ARRAY JA     149
C  CONTAINS THE COLUMN NUMBERS CORRESPONDING TO THE NONZEROES STORED   15Ø
C  IN THE ARRAY A (I.E., IF THE NONZERO STORED IN A(K) IS IN           151
C  COLUMN J, THEN JA(K) = J).  THE ARRAY IA CONTAINS POINTERS TO THE   152
C  ROWS OF NONZEROES/COLUMN INDICES IN THE ARRAY A/JA (I.E.,           153
C  A(IA(I))/JA(IA(I)) IS THE FIRST ENTRY FOR ROW I IN THE ARRAY A/JA).  154
C  IA(N+1) IS SET SO THAT IA(N+1) - IA(1) = THE NUMBER OF NONZEROES IN 155
C  A.  IU, JU, AND U ARE USED IN A SIMILAR WAY TO STORE THE STRICT UPPER 156
C  TRIANGLE OF U, EXCEPT THAT JU ACTUALLY CONTAINS C(J) INSTEAD OF J   157
C                                                                      158
C                                                                      159
      REAL A(1),B(1),U(1),X(1),Y(1)                                    16Ø
      REAL DK,LKI,ONE,XPV,XPVMAX,YK,ZERO                               161
      INTEGER C(1),IA(1),IC(1),IU(1),JA(1),JU(1),P(1),R(1)             162
      INTEGER CK,PK,PPK,PV,V,VI,VJ,VK                                  163
C                                                                      164
C                                                                      165
      IF (N .EQ. Ø) GO TO 1ØØ1                                         166
C                                                                      167
      ONE = 1.Ø                                                        168
      ZERO = Ø.Ø                                                       169
C                                                                      17Ø
```

```
C  INITIALIZE WORK STORAGE AND POINTERS TO JU                          171
C                                                                     172
      DO 1Ø J=1,N                                                     173
        X(J) = ZERO                                                   174
 1Ø     CONTINUE                                                      175
      IU(1) = 1                                                       176
      JUPTR = Ø                                                       177
C                                                                     178
C  PERFORM SYMBOLIC AND NUMERIC FACTORIZATION ROW BY ROW              179
C  VK (VI,VJ) IS THE GRAPH VERTEX FOR ROW K (I,J) OF U                18Ø
C                                                                     181
      DO 17Ø K=1,N                                                    182
C                                                                     183
C  INITIALIZE LINKED LIST AND FREE STORAGE FOR THIS ROW               184
C  THE R(K)-TH ROW OF A BECOMES THE K-TH ROW OF U.                    185
C                                                                     186
        P(N+1) = N+1                                                  187
        VK = R(K)                                                     188
C                                                                     189
C  SET UP ADJACENCY LIST FOR VK, ORDERED IN                           19Ø
C  CURRENT COLUMN ORDER OF U.  THE LOOP INDEX                         191
C  GOES DOWNWARD TO EXPLOIT ANY COLUMNS                               192
C  FROM A IN CORRECT RELATIVE ORDER                                   193
C                                                                     194
        JMIN = IA(VK)                                                 195
        JMAX = IA(VK+1) - 1                                           196
        IF (JMIN .GT. JMAX)  GO TO 1ØØ2                               197
        J = JMAX                                                      198
 2Ø       JAJ = JA(J)                                                 199
          VJ = IC(JAJ)                                                2ØØ
C                                                                     2Ø1
C  STORE A(K,J) IN WORK VECTOR                                        2Ø2
C                                                                     2Ø3
          X(VJ) = A(J)                                                2Ø4
C  THIS CODE INSERTS VJ INTO ADJACENCY LIST OF VK                     2Ø5
          PPK = N+1                                                   2Ø6
 3Ø       PK = PPK                                                    2Ø7
          PPK = P(PK)                                                 2Ø8
          IF (PPK - VJ)  3Ø,1ØØ3,4Ø                                   2Ø9
 4Ø       P(VJ) = PPK                                                 21Ø
          P(PK) = VJ                                                  211
          J = J - 1                                                   212
          IF (J .GE. JMIN) GO TO 2Ø                                   213
C                                                                     214
C  THE FOLLOWING CODE COMPUTES THE K-TH ROW OF U                      215
C                                                                     216
        VI = N+1                                                      217
        YK = B(VK)                                                    218
 5Ø     VI = P(VI)                                                    219
        IF (VI .GE. K) GO TO 11Ø                                      22Ø
C                                                                     221
C  VI LT VK -- PROCESS THE L(K,I) ELEMENT AND MERGE THE               222
C  ADJACENCY OF VI WITH THE ORDERED ADJACENCY OF VK                   223
C                                                                     224
        LKI = - X(VI)                                                 225
        X(VI) = ZERO                                                  226
C                                                                     227
C  ADJUST RIGHT HAND SIDE TO REFLECT ELIMINATION                      228
C                                                                     229
        YK = YK + LKI * Y(VI)                                         23Ø
        PPK = VI                                                      231
        JMIN = IU(VI)                                                 232
        JMAX = IU(VI+1) - 1                                           233
        IF (JMIN .GT. JMAX)  GO TO 5Ø                                 234
        DO 1ØØ J=JMIN,JMAX                                            235
          JUJ = JU(J)                                                 236
          VJ = IC(JUJ)                                                237
C                                                                     238
C  IF VJ IS ALREADY IN THE ADJACENCY OF VK,                           239
C  SKIP THE INSERTION                                                 24Ø
C                                                                     241
          IF (X(VJ) .NE. ZERO)  GO TO 9Ø                              242
C                                                                     243
C  INSERT VJ IN ADJACENCY LIST OF VK.                                 244
C  RESET PPK TO VI IF WE HAVE PASSED THE CORRECT                      245
C  INSERTION SPOT.  (THIS HAPPENS WHEN THE ADJACENCY OF               246
```

```
C  VI IS NOT IN CURRENT COLUMN ORDER DUE TO PIVOTING.)                    247
C                                                                        248
           IF (VJ - PPK) 6Ø,9Ø,7Ø                                        249
 6Ø        PPK = VI                                                      25Ø
 7Ø        PK = PPK                                                      251
           PPK = P(PK)                                                   252
           IF (PPK - VJ)  7Ø,9Ø,8Ø                                       253
 8Ø        P(VJ) = PPK                                                   254
           P(PK) = VJ                                                    255
           PPK = VJ                                                      256
C                                                                        257
C  COMPUTE L(K,J) = L(K,J) - L(K,I)*U(I,J) FOR L(K,I) NONZERO            258
C  COMPUTE U*(K,J) = U*(K,J) - L(K,I)*U(I,J) FOR U(K,J) NONZERO          259
C  (U*(K,J) = U(K,J)*D(K,K))                                             26Ø
C                                                                        261
 9Ø        X(VJ) = X(VJ) + LKI * U(J)                                    262
 1ØØ       CONTINUE                                                      263
         GO TO 5Ø                                                        264
C                                                                        265
C  PIVOT--INTERCHANGE LARGEST ENTRY OF K-TH ROW OF U WITH                 266
C  THE DIAGONAL ENTRY.                                                   267
C                                                                        268
C  FIND LARGEST ENTRY, COUNTING OFF-DIAGONAL NONZEROES                   269
C                                                                        27Ø
 11Ø     IF (VI .GT. N) GO TO 1ØØ4                                       271
         XPVMAX = ABS(X(VI))                                             272
         MAXC = VI                                                       273
         NZCNT = Ø                                                       274
         PV = VI                                                         275
 12Ø       V = PV                                                        276
           PV = P(PV)                                                    277
           IF (PV .GT. N) GO TO 13Ø                                      278
           NZCNT = NZCNT + 1                                             279
           XPV = ABS(X(PV))                                              28Ø
           IF (XPV .LE. XPVMAX) GO TO 12Ø                                281
           XPVMAX = XPV                                                  282
           MAXC = PV                                                     283
           MAXCL = V                                                     284
           GO TO 12Ø                                                     285
 13Ø     IF (XPVMAX .EQ. ZERO) GO TO 1ØØ4                                286
C                                                                        287
C  IF VI = K, THEN THERE IS AN ENTRY FOR DIAGONAL                        288
C  WHICH MUST BE DELETED.  OTHERWISE, DELETE THE                         289
C  ENTRY WHICH WILL BECOME THE DIAGONAL ENTRY                            29Ø
C                                                                        291
         IF (VI .EQ. K) GO TO 14Ø                                        292
         IF (VI .EQ. MAXC) GO TO 14Ø                                     293
         P(MAXCL) = P(MAXC)                                              294
         GO TO 15Ø                                                       295
 14Ø     VI = P(VI)                                                      296
C                                                                        297
C  COMPUTE D(K) = 1/L(K,K) AND PERFORM INTERCHANGE.                      298
C                                                                        299
 15Ø     DK = ONE / X(MAXC)                                              3ØØ
         X(MAXC) = X(K)                                                  3Ø1
         I = C(K)                                                        3Ø2
         C(K) = C(MAXC)                                                  3Ø3
         C(MAXC) = I                                                     3Ø4
         CK = C(K)                                                       3Ø5
         IC(CK) = K                                                      3Ø6
         IC(I) = MAXC                                                    3Ø7
         X(K) = ZERO                                                     3Ø8
C                                                                        3Ø9
C  UPDATE RIGHT HAND SIDE.                                               31Ø
C                                                                        311
         Y(K) = YK * DK                                                  312
C                                                                        313
C  COMPUTE VALUE FOR IU(K+1) AND CHECK FOR STORAGE OVERFLOW              314
C                                                                        315
         IU(K+1) = IU(K) + NZCNT                                         316
         IF (IU(K+1) .GT. MAX+1) GO TO 1ØØ5                              317
C                                                                        318
C  MOVE COLUMN INDICES FROM LINKED LIST TO JU.                           319
C  COLUMNS ARE STORED IN CURRENT ORDER WITH ORIGINAL                     32Ø
C  COLUMN NUMBER (C(J)) STORED FOR CURRENT COLUMN J                      321
C                                                                        322
```

```
      IF (VI .GT. N)  GO TO 17Ø                                          323
      J = VI                                                             324
 16Ø    JUPTR = JUPTR + 1                                               325
        JU(JUPTR) = C(J)                                                326
        U(JUPTR) = X(J) * DK                                            327
        X(J) = ZERO                                                     328
        J = P(J)                                                        329
        IF (J .LE. N) GO TO 16Ø                                         33Ø
 17Ø  CONTINUE                                                          331
C                                                                       332
C  BACKSOLVE U X = Y, AND REORDER X TO CORRESPOND WITH A                333
C                                                                       334
    K = N                                                               335
    DO 2ØØ I=1,N                                                        336
      YK = Y(K)                                                         337
      JMIN = IU(K)                                                      338
      JMAX = IU(K+1) - 1                                                339
      IF (JMIN .GT. JMAX)  GO TO 19Ø                                    34Ø
      DO 18Ø J=JMIN,JMAX                                                341
        JUJ = JU(J)                                                     342
        JUJ = IC(JUJ)                                                   343
        YK = YK - U(J) * Y(JUJ)                                         344
 18Ø    CONTINUE                                                        345
 19Ø  Y(K) = YK                                                         346
      CK = C(K)                                                         347
      X(CK) = YK                                                        348
      K = K-1                                                           349
 2ØØ  CONTINUE                                                          35Ø
C                                                                       351
C  RETURN WITH IERR = NUMBER OF OFF-DIAGONAL NONZEROES IN U             352
C                                                                       353
    IERR = IU(N+1) - IU(1)                                              354
    RETURN                                                              355
C                                                                       356
C  ERROR RETURNS                                                        357
C                                                                       358
C  N = Ø                                                                359
C                                                                       36Ø
 1ØØ1 IERR = Ø                                                          361
    RETURN                                                              362
C                                                                       363
C  ROW K OF A IS NULL                                                   364
C                                                                       365
 1ØØ2 IERR = -K                                                         366
    RETURN                                                              367
C                                                                       368
C  ROW K OF A HAS A DUPLICATE ENTRY                                     369
C                                                                       37Ø
 1ØØ3 IERR = -(N+K)                                                     371
    RETURN                                                              372
C                                                                       373
C  ZERO PIVOT IN ROW K                                                  374
C                                                                       375
 1ØØ4 IERR = -(2*N+K)                                                   376
    RETURN                                                              377
C                                                                       378
C  STORAGE FOR U EXCEEDED ON ROW K                                      379
C                                                                       38Ø
 1ØØ5 IERR = -(3*N+K)                                                   381
    RETURN                                                              382
    END                                                                 383
```

APPENDIX A

The code in this appendix illustrates the use of NSPIV. The linear system is of the form

$$Ax = b$$

where A is a 10×10 block tridiagonal matrix with 10×10 blocks. Specifically, This example is chosen for its simplicity; it does not exercise the algorithm, since A is a strictly diagonally dominant matrix.

$$A = \begin{pmatrix} C & D & & & \\ B & C & D & & \\ & B & \ddots & \ddots & \\ & & \ddots & \ddots & D \\ & & & B & C \end{pmatrix}$$

with

$$B = \begin{pmatrix} -1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{pmatrix}, \quad C = \begin{pmatrix} 4 & & & \\ -1 & 4 & & \\ & -1 & \ddots & \\ & & \ddots & 4 \end{pmatrix}$$

$$D = \begin{pmatrix} -1.5 & & & \\ & -1.5 & & \\ & & \ddots & \\ & & & -1.5 \end{pmatrix}.$$

```
C     PROGRAM PIVCHK(OUTPUT,TAPE6=OUTPUT)                                1
C                                                                        2
C  THIS PROGRAM ILLUSTRATES THE USE OF NSPIV BY SOLVING THE              3
C  SYSTEM OF LINEAR EQUATIONS                                            4
C                                                                        5
C       A X = B                                                          6
C                                                                        7
C  WITH A AN NG X NG BLOCK TRIDIAGONAL MATRIX, WITH NG X NG BLOCKS.      8
C  THE DIAGONAL BLOCKS OF A ARE LOWER BI-DIAGONAL (ENTRIES ARE 4.Ø       9
C  ON THE DIAGONAL, -1.Ø ON THE SUBDIAGONAL), AND THE OFF-DIAGONAL       1Ø
C  BLOCKS OF A ARE DIAGONAL (ENTRIES ARE -1.Ø IN THE LOWER TRIANGLE,     11
C  -1.5 IN THE UPPER TRIANGLE.)  X IS CHOSEN TO BE A VECTOR              12
C  OF ALL ONES, AND B IS COMPUTED ACCORDINGLY.                           13
C                                                                        14
C                                                                        15
      INTEGER IA(1Ø1),JA(4ØØ),R(1ØØ),C(1ØØ),IC(1ØØ),ITEMP(597)           16
      REAL A(4ØØ),B(1ØØ),X(1ØØ),RTEMP(495)                              17
      DATA MAX/395/,NG/1Ø/,N/1ØØ/                                       18
C                                                                        19
C  SET UP PROBLEM                                                        2Ø
C                                                                        21
      K = 1                                                              22
      IA(1) = 1                                                          23
      IAPTR = 1                                                          24
      DO 5 I=1,NG                                                        25
        DO 5 J=1,NG                                                      26
          BK = Ø.                                                        27
          IF (I .EQ. 1) GO TO 1                                          28
          JA(IAPTR) = K - NG                                             29
          A(IAPTR) = -1.                                                 3Ø
          BK = BK - 1.                                                   31
          IAPTR = IAPTR + 1                                              32
 1        IF (J .EQ. 1) GO TO 2                                          33
          JA(IAPTR) = K - 1                                              34
          A(IAPTR) = -1.                                                 35
          BK = BK - 1.                                                   36
```

```
            IAPTR = IAPTR + 1                                          37
 2          JA(IAPTR) = K                                              38
            A(IAPTR) = 4.                                              39
            BK = BK + 4.                                               4Ø
            IAPTR = IAPTR + 1                                          41
            IF (I .EQ. NG) GO TO 4                                     42
            JA(IAPTR) = K + NG                                         43
            A(IAPTR) = -1.5                                            44
            BK = BK - 1.5                                              45
            IAPTR = IAPTR + 1                                          46
 4          B(K) = BK                                                  47
            K = K + 1                                                  48
            IA(K) = IAPTR                                              49
 5          CONTINUE                                                   5Ø
C                                                                       51
C  CALL PREORD TO ORDER ROWS OF A BY INCREASING NUMBERS OF NONZEROES    52
C                                                                       53
      CALL PREORD(N,IA,R,C,IC)                                          54
C                                                                       55
C  CALL NSPIV TO SOLVE SYSTEM                                           56
C                                                                       57
      CALL NSPIV(N,IA,JA,A,B,MAX,R,C,IC,X,ITEMP,RTEMP,IERR)             58
      WRITE (6,1Ø1) IERR                                                59
 1Ø1  FORMAT (8H IERR = ,I1Ø)                                           6Ø
C                                                                       61
C  CALL RESCHK TO COMPUTE MAX-NORM AND 2-NORM OF RESIDUAL               62
C                                                                       63
      CALL RESCHK(N,IA,JA,A,B,X)                                        64
C                                                                       65
      STOP                                                              66
      END                                                               67

      SUBROUTINE PREORD(N,IA,R,C,IC)                                    68
C                                                                       69
C  PREORD ORDERS THE ROWS OF A BY INCREASING NUMBER OF NONZEROES.       7Ø
C  THE ROW PERMUTATION IS RETURNED IN R.  C IS SET TO THE IDENTITY.     71
C                                                                       72
      INTEGER IA(1),R(1),C(1),IC(1)                                     73
C                                                                       74
      DO 1 I=1,N                                                        75
        R(I) = I                                                        76
        C(I) = I                                                        77
        IC(I) = I                                                       78
1       CONTINUE                                                        79
      DO 5 I = 1,N                                                      8Ø
5       C(I) = Ø                                                        81
      DO 1Ø K = 1,N                                                     82
        KDEG = IA(K+1) - IA(K)                                          83
        IF (KDEG .EQ. Ø) KDEG = KDEG + 1                                84
        IC(K) = C(KDEG)                                                 85
        C(KDEG) = K                                                     86
1Ø      CONTINUE                                                        87
      I = Ø                                                             88
      DO 3Ø J = 1,N                                                     89
        IF (C(J) .EQ. Ø) GO TO 3Ø                                       9Ø
        K = C(J)                                                        91
2Ø      I = I + 1                                                       92
        R(I) = K                                                        93
        K = IC(K)                                                       94
        IF (K .GT. Ø) GO TO 2Ø                                          95
3Ø      CONTINUE                                                        96
      DO 4Ø I = 1,N                                                     97
        C(I) = I                                                        98
        IC(I) = I                                                       99
4Ø      CONTINUE                                                       1ØØ
      RETURN                                                           1Ø1
      END                                                              1Ø2
      SUBROUTINE RESCHK(N,IA,JA,A,B,X)                                 1Ø3
C                                                                      1Ø4
C  RESCHK COMPUTES THE MAX-NORM AND 2-NORM OF THE RESIDUAL.            1Ø5
C  DOUBLE PRECISION IS USED FOR THE COMPUTATION.                       1Ø6
C                                                                      1Ø7
      INTEGER IA(1),JA(1)                                              1Ø8
      REAL A(1),B(1),X(1)                                              1Ø9
      DOUBLE PRECISION RESID,RESIDM,ROWSUM                             11Ø
      RESID = Ø.                                                       111
```

```
      RESIDM = ∅.                                                          112
      DO 2∅ I=1,N                                                          113
        ROWSUM = DBLE(B(I))                                                114
        JMIN = IA(I)                                                       115
        JMAX = IA(I+1) - 1                                                 116
        DO 1∅ J=JMIN,JMAX                                                  117
          JAJ = JA(J)                                                      118
          ROWSUM = ROWSUM - DBLE(A(J)) * DBLE(X(JAJ))                      119
1∅      CONTINUE                                                           12∅
        IF (DABS(ROWSUM) .GT. RESIDM) RESIDM = DABS(ROWSUM)                121
        RESID = RESID + ROWSUM**2                                          122
2∅      CONTINUE                                                           123
      RESID = DSQRT(RESID)                                                 124
      WRITE (6,25) RESID                                                   125
25    FORMAT (22H 2-NORM OF RESIDUAL = ,D14.7)                             126
      WRITE (6,3∅) RESIDM                                                  127
3∅    FORMAT(24H MAX NORM OF RESIDUAL = ,D14.7)                            128
      RETURN                                                               129
      END                                                                  13∅


C     PROGRAM PIVCHK(INPUT,OUTPUT,TAPE6=OUTPUT,TAPE5=INPUT)                 1
C                                                                           2
C  THIS PROGRAM ILLUSTRATES THE USE OF NSPIV BY SOLVING THE                 3
C  SYSTEM OF LINEAR EQUATIONS                                               4
C                                                                           5
C       A X = B                                                             6
C                                                                           7
C  WHERE A IS A BANDED 192 X 192 MATRIX OF BANDWIDTH 2∅.  X IS              8
C  CHOSEN TO BE A VECTOR OF ALL ONES, AND B IS COMPUTED ACCORDINGLY.        9
C                                                                          1∅
C                                                                          11
      INTEGER IA(193),JA(35∅∅),R(192),C(192),IC(192),ITEMP(75∅∅)           12
      REAL A(35∅∅),B(192),X(192),RTEMP(72∅∅)                               13
      DATA MAX/7∅∅∅/,N/192/                                                14
C                                                                          15
C  CALL GENPRB TO SET UP PROBLEM                                           16
C                                                                          17
      CALL GENPRB(N,IA,JA,A,ITEMP,B)                                       18
C                                                                          19
C  CALL PREORD TO ORDER ROWS OF A BY INCREASING NUMBERS OF NONZEROES        2∅
C                                                                          21
      CALL PREORD(N,IA,R,C,IC)                                             22
C                                                                          23
C  CALL NSPIV TO SOLVE SYSTEM                                              24
C                                                                          25
      CALL NSPIV(N,IA,JA,A,B,MAX,R,C,IC,X,ITEMP,RTEMP,IERR)                26
      WRITE (6,1∅1) IERR                                                   27
 1∅1  FORMAT (8H IERR = ,I1∅)                                              28
C                                                                          29
C  CALL RESCHK TO COMPUTE MAX-NORM AND 2-NORM OF RESIDUAL                  3∅
C                                                                          31
      CALL RESCHK(N,IA,JA,A,B,X)                                           32
C                                                                          33
      STOP                                                                 34
      END                                                                  35
      SUBROUTINE GENPRB(N,IA,JA,A,B,RHS)                                  1∅4
C                                                                         1∅5
C  GENPRB SETS UP THE MATRIX AND RIGHT HAND SIDE FROM                     1∅6
C  THE DATA ON CARDS.  THE PROBLEM IS ONE FROM THE USGS.                  1∅7
C                                                                         1∅8
      INTEGER IA(1),JA(1)                                                 1∅9
      REAL B(1),A(1),RHS(1)                                               11∅
C                                                                         111
      KMIN = 1                                                            112
      DO 5 I=1,N                                                          113
        KMAX = KMIN + 38                                                  114
        READ (5,1∅1) (B(K),K=KMIN,KMAX)                                   115
 1∅1    FORMAT (8G1∅.3)                                                   116
        KMIN = KMAX + 1                                                   117
 5      CONTINUE                                                          118
C                                                                         119
      IA(1) = 1                                                           12∅
      IAPTR = 1                                                           121
      IPART = ∅                                                           122
      DO 15 I=1,N                                                         123
        RHSI = ∅.                                                         124
```

```
         DO 1Ø J=1,39
           K = I - 2Ø + J
           IF (K .LE. Ø) GO TO 1Ø
           IF (K .GT. N) GO TO 1Ø
           IPARTJ = IPART + J
           IF (B(IPARTJ) .EQ. Ø.) GO TO 1Ø
           A(IAPTR) = B(IPARTJ)
           JA(IAPTR) = K
           RHSI = RHSI + B(IAPTR)
           IAPTR = IAPTR + 1
1Ø         CONTINUE
         IA(I+1) = IAPTR
         RHS(I) = RHSI
15       IPART = IPART + 39
       RETURN
       END
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø          Ø  2.67      -.819      -.475      -.4Ø9              Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø          Ø -.475      -.4Ø9      -1.14      -.2Ø4
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø 1.96ØE-Ø2  .125      -7.451E-Ø3 6.258E-Ø2          Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø-7.451E-Ø3 6.258E-Ø2-1.233E-Ø2 3.117E-Ø2          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø-2.97ØE-Ø2-2.556E-Ø2  .333      -.1Ø2     -2.97ØE-Ø2-2.556E-Ø2          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-7.451E-Ø3 6.258E-Ø2 3.912E-Ø2  .249      -7.451E-Ø3 6.258E-Ø2          Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-1.233E-Ø2 3.117E-Ø2-1.49ØE-Ø2  .125      -1.233E-Ø2 3.117E-Ø2          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø-2.97ØE-Ø2-2.556E-Ø2  .333      -.1Ø2     -2.97ØE-Ø2-2.556E-Ø2          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
  -.119       1.ØØ       .626       3.99      -.119       1.ØØ              Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
  -.197       .499      -.238       1.99      -.197       .499              Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø-2.97ØE-Ø2-2.556E-Ø2  .333      -.1Ø2     -2.97ØE-Ø2-2.556E-Ø2          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
  -.119       1.ØØ       .626       3.99      -.119       1.ØØ              Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
  -.197       .499      -.238       1.99      -.197       .499              Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø-2.97ØE-Ø2-2.556E-Ø2  .333      -.1Ø2     -2.97ØE-Ø2-2.556E-Ø2          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
  -.119       1.ØØ       .626       3.99      -.119       1.ØØ              Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
  -.197       .499      -.238       1.99      -.197       .499              Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø-2.97ØE-Ø2-2.556E-Ø2  .333      -.1Ø2     -2.97ØE-Ø2-2.556E-Ø2          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
  -.119       1.ØØ       .626       3.99      -.119       1.ØØ              Ø          Ø
         Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
  -.197       .499      -.238       1.99      -.197       .499              Ø
```

```
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-2.97ØE-Ø2-2.556E-Ø2  .333     -.1Ø2    -2.97ØE-Ø2-2.556E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-7.451E-Ø3 6.258E-Ø2 3.912E-Ø2  .249    -7.451E-Ø3 6.258E-Ø2         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-1.233E-Ø2 3.117E-Ø2-1.49ØE-Ø2  .125    -1.233E-Ø2 3.117E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø -.475     -.4Ø9      2.67     -.819             Ø         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø -1.14     -.2Ø4     -.475     -.4Ø9             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-7.451E-Ø3 6.258E-Ø2 1.96ØE-Ø2  .125             Ø         Ø         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-1.233E-Ø2 3.117E-Ø2-7.451E-Ø3 6.258E-Ø2         Ø         Ø         Ø
         Ø         Ø         Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø         Ø  .333     -.1Ø2    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø         Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2
         Ø         Ø-7.451E-Ø3 6.258E-Ø2-1.233E-Ø2 3.117E-Ø2         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø 3.912E-Ø2  .249    -1.49ØE-Ø2  .125             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø-7.451E-Ø3 6.258E-Ø2-1.233E-Ø2 3.117E-Ø2         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-1.233E-Ø2 3.117E-Ø2-1.49ØE-Ø2  .125    -1.233E-Ø2 3.117E-Ø2         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-1.49ØE-Ø2  .125     7.823E-Ø2  .499    -1.49ØE-Ø2  .125             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-1.233E-Ø2 3.117E-Ø2-1.49ØE-Ø2  .125    -1.233E-Ø2 3.117E-Ø2         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -.197      .499     -.238      1.99     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.238      1.99     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -.197      .499     -.238      1.99     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.238      1.99     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -.197      .499     -.238      1.99     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.238      1.99     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -.197      .499     -.238      1.99     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
```

```
 -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.238      1.99     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-1.233E-Ø2 3.117E-Ø2-1.49ØE-Ø2  .125    -1.233E-Ø2 3.117E-Ø2         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-1.49ØE-Ø2  .125     7.823E-Ø2  .499    -1.49ØE-Ø2  .125             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-1.233E-Ø2 3.117E-Ø2-1.49ØE-Ø2  .125    -1.233E-Ø2 3.117E-Ø2         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2         Ø         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .333     -.1Ø2             Ø         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2         Ø         Ø
-1.233E-Ø2 3.117E-Ø2-7.451E-Ø3 6.258E-Ø2         Ø         Ø         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-1.49ØE-Ø2  .125     3.912E-Ø2  .249             Ø         Ø         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-1.233E-Ø2 3.117E-Ø2-7.451E-Ø3 6.258E-Ø2         Ø         Ø         Ø
         Ø         Ø         Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø         Ø  .333     -.1Ø2    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø         Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2
         Ø         Ø -.119      1.ØØ     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø  .626      3.99     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø -.119      1.ØØ     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -.197      .499     -.238      1.99     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.238      1.99     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -.197      .499     -.238      1.99     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.238      1.99     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -.197      .499     -.238      1.99     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.238      1.99     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -.197      .499     -.238      1.99     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.238      1.99     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
```

```
          Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
  -.197      .499    -.238      1.99     -.197      .499                Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.238      1.99      1.25      7.98     -.238      1.99                Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.197      .499    -.238      1.99     -.197      .499                Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
  -.197      .499    -.238      1.99     -.197      .499                Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.238      1.99      1.25      7.98     -.238      1.99                Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.197      .499    -.238      1.99     -.197      .499                Ø
          Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2        Ø         Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-5.939E-Ø2-5.12ØE-Ø2  .333     -.1Ø2            Ø         Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2        Ø         Ø
  -.197      .499    -.119      1.ØØ            Ø         Ø         Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.238      1.99      .626      3.99            Ø         Ø         Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.197      .499    -.119      1.ØØ            Ø         Ø         Ø
          Ø         Ø         Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø         Ø         Ø  .333     -.1Ø2    -5.939E-Ø2-5.12ØE-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø         Ø         Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2
          Ø         Ø -.119      1.ØØ     -.197      .499                Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø         Ø  .626      3.99     -.238      1.99                Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø         Ø -.119      1.ØØ     -.197      .499                Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
  -.197      .499    -.238      1.99     -.197      .499                Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.238      1.99      1.25      7.98     -.238      1.99                Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.197      .499    -.238      1.99     -.197      .499                Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
  -.197      .499    -.238      1.99     -.197      .499                Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.238      1.99      1.25      7.98     -.238      1.99                Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.197      .499    -.238      1.99     -.197      .499                Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
  -.197      .499    -.238      1.99     -.197      .499                Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.238      1.99      1.25      7.98     -.238      1.99                Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.197      .499    -.238      1.99     -.197      .499                Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
  -.197      .499    -.238      1.99     -.197      .499                Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.238      1.99      1.25      7.98     -.238      1.99                Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
```

```
-.197      .499      -.238       1.99      -.197       .499              Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238       1.99      -.197       .499              Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       1.25       7.98      -.238       1.99              Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238       1.99      -.197       .499              Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238       1.99      -.197       .499              Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       1.25       7.98      -.238       1.99              Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238       1.99      -.197       .499              Ø
        Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2            Ø          Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .333      -.1Ø2                 Ø          Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2            Ø          Ø
-.197      .499      -.119       1.ØØ                  Ø          Ø          Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       .626       3.99                  Ø          Ø          Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.119       1.ØØ                  Ø          Ø          Ø
        Ø         Ø          Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø         Ø          Ø  .333      -.1Ø2    -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø         Ø          Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2
        Ø         Ø -.119       1.ØØ      -.197       .499              Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø         Ø  .626       3.99      -.238       1.99              Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø         Ø -.119       1.ØØ      -.197       .499              Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238       1.99      -.197       .499              Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       1.25       7.98      -.238       1.99              Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238       1.99      -.197       .499              Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238       1.99      -.197       .499              Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       1.25       7.98      -.238       1.99              Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238       1.99      -.197       .499              Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238       1.99      -.197       .499              Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       1.25       7.98      -.238       1.99              Ø          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238       1.99      -.197       .499              Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø         Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238       1.99      -.197       .499              Ø          Ø
```

```
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.238      1.99     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -.197      .499     -.238      1.99     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.238      1.99     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -.197      .499     -.238      1.99     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.238      1.99     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2         Ø         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .333     -.1Ø2             Ø         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2         Ø         Ø
 -.197      .499     -.119      1.ØØ             Ø         Ø         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.238      1.99      .626      3.99             Ø         Ø         Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.119      1.ØØ             Ø         Ø         Ø
         Ø         Ø         Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø         Ø  .333     -.1Ø2    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø         Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2
         Ø         Ø -.119      1.ØØ     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø  .626      3.99     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø         Ø -.119      1.ØØ     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -.197      .499     -.238      1.99     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.238      1.99     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -.197      .499     -.238      1.99     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.238      1.99     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -.197      .499     -.238      1.99     -.197      .499             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
 -.197      .499     -.238      1.99     -.197      .499             Ø
         Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
         Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
         Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
```

```
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238      1.99      -.197      .499                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       1.25       7.98      -.238      1.99                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238      1.99      -.197      .499                Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238      1.99      -.197      .499                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       1.25       7.98      -.238      1.99                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238      1.99      -.197      .499                Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238      1.99      -.197      .499                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       1.25       7.98      -.238      1.99                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238      1.99      -.197      .499                Ø
        Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .333      -.1Ø2               Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2          Ø          Ø
-.197      .499      -.119      1.ØØ                Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       .626       3.99               Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.119      1.ØØ                Ø          Ø          Ø
        Ø          Ø          Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø  .333      -.1Ø2     -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2
        Ø          Ø -.119      1.ØØ      -.197      .499                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø  .626      3.99      -.238      1.99                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø -.119      1.ØØ      -.197      .499                Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238      1.99      -.197      .499                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       1.25       7.98      -.238      1.99                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238      1.99      -.197      .499                Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238      1.99      -.197      .499                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       1.25       7.98      -.238      1.99                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238      1.99      -.197      .499                Ø
```

```
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499     -.238      1.99      -.197      .499            Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
-.238      1.99      1.25      7.98      -.238      1.99            Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
-.197      .499     -.238      1.99      -.197      .499            Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499     -.238      1.99      -.197      .499            Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
-.238      1.99      1.25      7.98      -.238      1.99            Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
-.197      .499     -.238      1.99      -.197      .499            Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499     -.238      1.99      -.197      .499            Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
-.238      1.99      1.25      7.98      -.238      1.99            Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
-.197      .499     -.238      1.99      -.197      .499            Ø
        Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2           Ø         Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .333      -.1Ø2                Ø         Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2           Ø         Ø
-.197      .499     -.119      1.ØØ                Ø         Ø         Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
-.238      1.99      .626      3.99                Ø         Ø         Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
-.197      .499     -.119      1.ØØ                Ø         Ø         Ø
        Ø         Ø         Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø         Ø         Ø  .333      -.1Ø2     -5.939E-Ø2-5.12ØE-Ø2        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø         Ø         Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2
        Ø         Ø -.119      1.ØØ      -.197      .499            Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø         Ø  .626      3.99      -.238      1.99            Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø         Ø -.119      1.ØØ      -.197      .499            Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499     -.238      1.99      -.197      .499            Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
-.238      1.99      1.25      7.98      -.238      1.99            Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
-.197      .499     -.238      1.99      -.197      .499            Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499     -.238      1.99      -.197      .499            Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
-.238      1.99      1.25      7.98      -.238      1.99            Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
-.197      .499     -.238      1.99      -.197      .499            Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499     -.238      1.99      -.197      .499            Ø        Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø        Ø
```

```
-.238      1.99      1.25      7.98     -.238     1.99              Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-.197      .499     -.238      1.99     -.197      .499             Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499     -.238      1.99     -.197      .499             Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-.238      1.99      1.25      7.98     -.238     1.99              Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-.197      .499     -.238      1.99     -.197      .499             Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499     -.238      1.99     -.197      .499             Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-.238      1.99      1.25      7.98     -.238     1.99              Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-.197      .499     -.238      1.99     -.197      .499             Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499     -.238      1.99     -.197      .499             Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-.238      1.99      1.25      7.98     -.238     1.99              Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-.197      .499     -.238      1.99     -.197      .499             Ø
        Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2         Ø         Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .333     -.1Ø2             Ø         Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2         Ø         Ø
-.197      .499     -.119      1.ØØ             Ø         Ø         Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-.238      1.99      .626      3.99             Ø         Ø         Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-.197      .499     -.119      1.ØØ             Ø         Ø         Ø
        Ø         Ø         Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø         Ø         Ø  .333     -.1Ø2    -5.939E-Ø2-5.12ØE-Ø2         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø         Ø         Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2
        Ø         Ø -.119      1.ØØ     -.197      .499             Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø         Ø  .626      3.99     -.238     1.99              Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø         Ø -.119      1.ØØ     -.197      .499             Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499     -.238      1.99     -.197      .499             Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-.238      1.99      1.25      7.98     -.238     1.99              Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-.197      .499     -.238      1.99     -.197      .499             Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499     -.238      1.99     -.197      .499             Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-.238      1.99      1.25      7.98     -.238     1.99              Ø         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
-.197      .499     -.238      1.99     -.197      .499             Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
        Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
```

```
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
  -.197      .499    -.238       1.99     -.197      .499             Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.197      .499    -.238       1.99     -.197      .499             Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
  -.197      .499    -.238       1.99     -.197      .499             Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.197      .499    -.238       1.99     -.197      .499             Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
  -.197      .499    -.238       1.99     -.197      .499             Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.197      .499    -.238       1.99     -.197      .499             Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
  -.197      .499    -.238       1.99     -.197      .499             Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.197      .499    -.238       1.99     -.197      .499             Ø
          Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2         Ø         Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-5.939E-Ø2-5.12ØE-Ø2  .333     -.1Ø2             Ø         Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2         Ø         Ø
  -.197      .499    -.119       1.ØØ             Ø         Ø         Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.238      1.99      .626      3.99             Ø         Ø         Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.197      .499    -.119       1.ØØ             Ø         Ø         Ø
          Ø         Ø         Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø         Ø         Ø  .333     -.1Ø2    -5.939E-Ø2-5.12ØE-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø         Ø         Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2
          Ø         Ø -.119      1.ØØ     -.197      .499             Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø         Ø  .626      3.99     -.238      1.99             Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø         Ø -.119      1.ØØ     -.197      .499             Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
  -.197      .499    -.238       1.99     -.197      .499             Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.197      .499    -.238       1.99     -.197      .499             Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
          Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
  -.197      .499    -.238       1.99     -.197      .499             Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.238      1.99      1.25      7.98     -.238      1.99             Ø         Ø
          Ø         Ø         Ø         Ø         Ø         Ø         Ø         Ø
  -.197      .499    -.238       1.99     -.197      .499             Ø
```

```
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238      1.99      -.197      .499                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       1.25      7.98      -.238      1.99                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238      1.99      -.197      .499                Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238      1.99      -.197      .499                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       1.25      7.98      -.238      1.99                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238      1.99      -.197      .499                Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238      1.99      -.197      .499                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       1.25      7.98      -.238      1.99                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238      1.99      -.197      .499                Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-.197      .499      -.238      1.99      -.197      .499                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       1.25      7.98      -.238      1.99                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238      1.99      -.197      .499                Ø
        Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .333      -.1Ø2               Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2          Ø          Ø
-.197      .499      -.119      1.ØØ                Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.238      1.99       .626      3.99                Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.119      1.ØØ                Ø          Ø          Ø
        Ø          Ø          Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø  .333      -.1Ø2    -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø-2.97ØE-Ø2-2.556E-Ø2-7.142E-Ø2-1.278E-Ø2
        Ø          Ø-7.451E-Ø3 6.258E-Ø2-1.233E-Ø2 3.117E-Ø2          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø 3.912E-Ø2  .249    -1.49ØE-Ø2  .125                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø-7.451E-Ø3 6.258E-Ø2-1.233E-Ø2 3.117E-Ø2          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
-1.233E-Ø2 3.117E-Ø2-1.49ØE-Ø2  .125    -1.233E-Ø2 3.117E-Ø2          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-1.49ØE-Ø2  .125     7.823E-Ø2  .499    -1.49ØE-Ø2  .125                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-1.233E-Ø2 3.117E-Ø2-1.49ØE-Ø2  .125    -1.233E-Ø2 3.117E-Ø2          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-5.939E-Ø2-5.12ØE-Ø2  .668      -.2Ø5     -5.939E-Ø2-5.12ØE-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -.197      .499      -.238      1.99      -.197      .499                Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
```

```
   -.238     1.99     1.25     7.98    -.238     1.99              Ø        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
   -.197     .499    -.238     1.99    -.197     .499              Ø
            Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
            Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
            Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
   -.197     .499    -.238     1.99    -.197     .499              Ø        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
   -.238     1.99     1.25     7.98    -.238     1.99              Ø        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
   -.197     .499    -.238     1.99    -.197     .499              Ø
            Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
            Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
            Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
   -.197     .499    -.238     1.99    -.197     .499              Ø        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
   -.238     1.99     1.25     7.98    -.238     1.99              Ø        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
   -.197     .499    -.238     1.99    -.197     .499              Ø
            Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
            Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
            Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
   -.197     .499    -.238     1.99    -.197     .499              Ø        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
   -.238     1.99     1.25     7.98    -.238     1.99              Ø        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
   -.197     .499    -.238     1.99    -.197     .499              Ø
            Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
            Ø-5.939E-Ø2-5.12ØE-Ø2  .668     -.2Ø5    -5.939E-Ø2-5.12ØE-Ø2        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
            Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2
 -1.233E-Ø2 3.117E-Ø2-1.49ØE-Ø2  .125    -1.233E-Ø2 3.117E-Ø2          Ø        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
 -1.49ØE-Ø2  .125     7.823E-Ø2  .499    -1.49ØE-Ø2  .125              Ø        Ø
            Ø         Ø        Ø        Ø        Ø        Ø        Ø        Ø
 -1.233E-Ø2 3.117E-Ø2-1.49ØE-Ø2  .125    -1.233E-Ø2 3.117E-Ø2          Ø
           Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2         Ø        Ø        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø        Ø
           Ø-5.939E-Ø2-5.12ØE-Ø2  .333     -.1Ø2             Ø        Ø        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø        Ø
           Ø-7.142E-Ø2-1.278E-Ø2-2.97ØE-Ø2-2.556E-Ø2         Ø        Ø
 -1.233E-Ø2 3.117E-Ø2-7.451E-Ø3 6.258E-Ø2         Ø         Ø        Ø        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø        Ø
 -1.49ØE-Ø2  .125     3.912E-Ø2  .249              Ø         Ø        Ø        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø        Ø
 -1.233E-Ø2 3.117E-Ø2-7.451E-Ø3 6.258E-Ø2         Ø         Ø        Ø
           Ø         Ø        Ø -.475     -.4Ø9     -1.14     -.2Ø4               Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø        Ø
           Ø         Ø        Ø  2.67     -.819     -.475     -.4Ø9               Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø
           Ø         Ø-7.451E-Ø3 6.258E-Ø2-1.233E-Ø2 3.117E-Ø2         Ø        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø        Ø
           Ø         Ø 1.96ØE-Ø2  .125    -7.451E-Ø3 6.258E-Ø2         Ø        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø
           Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø        Ø
           Ø-2.97ØE-Ø2-2.556E-Ø2  .333     -.1Ø2    -2.97ØE-Ø2-2.556E-Ø2        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø
 -1.233E-Ø2 3.117E-Ø2-1.49ØE-Ø2  .125    -1.233E-Ø2 3.117E-Ø2          Ø        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø        Ø
 -7.451E-Ø3 6.258E-Ø2 3.912E-Ø2  .249    -7.451E-Ø3 6.258E-Ø2          Ø        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø
           Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø        Ø
           Ø-2.97ØE-Ø2-2.556E-Ø2  .333     -.1Ø2    -2.97ØE-Ø2-2.556E-Ø2        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø        Ø
           Ø         Ø        Ø        Ø        Ø         Ø        Ø
```

```
-.197      .499      -.238      1.99     -.197      .499              Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.119      1.ØØ       .626      3.99     -.119      1.ØØ              Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-2.97ØE-Ø2-2.556E-Ø2  .333      -.1Ø2    -2.97ØE-Ø2-2.556E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238      1.99     -.197      .499              Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.119      1.ØØ       .626      3.99     -.119      1.ØØ              Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-2.97ØE-Ø2-2.556E-Ø2  .333      -.1Ø2    -2.97ØE-Ø2-2.556E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238      1.99     -.197      .499              Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.119      1.ØØ       .626      3.99     -.119      1.ØØ              Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-2.97ØE-Ø2-2.556E-Ø2  .333      -.1Ø2    -2.97ØE-Ø2-2.556E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.197      .499      -.238      1.99     -.197      .499              Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-.119      1.ØØ       .626      3.99     -.119      1.ØØ              Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-7.142E-Ø2-1.278E-Ø2-5.939E-Ø2-5.12ØE-Ø2-7.142E-Ø2-1.278E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø-2.97ØE-Ø2-2.556E-Ø2  .333      -.1Ø2    -2.97ØE-Ø2-2.556E-Ø2          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
-1.233E-Ø2 3.117E-Ø2-1.49ØE-Ø2  .125     -1.233E-Ø2 3.117E-Ø2         Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-7.451E-Ø3 6.258E-Ø2 3.912E-Ø2  .249     -7.451E-Ø3 6.258E-Ø2         Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø -1.14      -.2Ø4      -.475      -.4Ø9                Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø -.475      -.4Ø9       2.67      -.819                Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
-1.233E-Ø2 3.117E-Ø2-7.451E-Ø3 6.258E-Ø2            Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
-7.451E-Ø3 6.258E-Ø2 1.96ØE-Ø2  .125                Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø          Ø
        Ø          Ø          Ø          Ø          Ø          Ø          Ø
```

ALGORITHM 534
STINT: STiff (differential equations) INTegrator [D2]

JOEL M. TENDLER
IBM Poughkeepsie Laboratories
THEODORE A. BICKART
Syracuse University
and
ZDENEK PICEL

Key Words and Phrases: stiff differential equations, stiffly stable methods, composite multistep methods, cyclic methods, numerical integration, ordinary differential equations, initial value problems, multistep formulas, numerical integration program, Fortran code STINT
CR Categories: 5.16, 5.17
Language: Fortran

DESCRIPTION

The algorithm given here is a complement to [1] where the theoretical development and test results are described.

REFERENCES

[1] TENDLER, J.M., BICKART, T.A., AND PICEL, Z. A stiffly stable integration process using cyclic composite methods. *ACM Trans. Math. Software 4*, 4 (Dec. 1978), 339-368.

ALGORITHM

```
C*                                                                    *MAIN  1Ø
C*---------- THIS IS A MAIN PROGRAM FOR  STINT-TYPE SUBROUTINES -------*MAIN  2Ø
C*--------------------- SET HERE FOR N = 4 PROBLEM - ------------------*MAIN  3Ø
C*                                                                    *MAIN  4Ø
      DOUBLE PRECISION D, EPS, ERO, ERORI, HI, HMAX, HMIN, HNEXT,      MAIN  5Ø
     1                 HØ, RJ, RW, S, SAVE, T, TF, TI, TOUT,          MAIN  6Ø
     2                 TOUTP, TS, YDOT, YI, YMAX, YOUT, YØ            MAIN  7Ø
C     DOUBLE PRECISION DABS, DBLE, DMAX1                              MAIN  8Ø
      DIMENSION YI(4), ERORI(4), YØ(4,8,4), YDOT(4,4), SAVE(52), RJ(16),MAIN  9Ø
     1          YMAX(4), RW(16), ERO(4), YOUT(4), IPIV(4)             MAIN 1ØØ
```

Received 14 April 1975, 2 May 1975, 15 December 1976, and 10 January 1978.

Authors' addresses: J. M. Tendler, IBM Corporation, Dept. C01, Bldg. 702, P.O. Box 390, Poughkeepsie, NY 12602; T. A. Bickart, Electrical and Computer Engineering Department, Link Hall, Syracuse University, Syracuse, NY 13210; Z. Picel, IBM France Scientific Center, 36 Av. R. Poincaré, 75016 Paris, France.

```
      COMMON /STAT/ NSTEP,NFE,NJE,NINVS                                  MAIN 11Ø
      DATA LIN/5/,LOUT/6/                                                MAIN 12Ø
C*                                                                       *MAIN 13Ø
C*       SET THE INPUT PARAMETERS                                        *MAIN 14Ø
C*                                                                       *MAIN 15Ø
C*       NP    PROBLEM IDENTIFIER                                        *MAIN 16Ø
C*       NC    NUMBER OF TIMES THE SAME PROBLEM IS SOLVED WITH DIFFERENT*MAIN 17Ø
C*             VALUE OF EPS                                              *MAIN 18Ø
C*       IP    NUMBER OF TEST POINTS AT WHICH THE SOLUTION IS SOUGHT     *MAIN 19Ø
C*             (DOES NOT INCLUDE THE FINAL POINT)                        *MAIN 2ØØ
C*       N     NUMBER OF DIFFERENTIAL EQUATIONS TO BE SOLVED             *MAIN 21Ø
C*       YI    THE INITIAL VALUES OF THE DEPENDENT VARIABLE              *MAIN 22Ø
C*       TI    THE INITIAL VALUE OF THE INDEPENDENT VARIABLE             *MAIN 23Ø
C*       TF    THE FINAL VALUE OF THE INDEPENDENT VARIABLE               *MAIN 24Ø
C*       HI    THE INITIAL STEP-SIZE                                     *MAIN 25Ø
C*       MF    METHOD FLAG DETERMINING THE MODE OF THE JACOBIAN          *MAIN 26Ø
C*             MATRIX EVALUATION                                         *MAIN 27Ø
C*       EPS   USERS  SPECIFIED ERROR PER STEP                           *MAIN 28Ø
C*                                                                       *MAIN 29Ø
      READ(LIN,1Ø) NP                                                    MAIN 3ØØ
      READ(LIN,1Ø) NC                                                    MAIN 31Ø
      READ(LIN,1Ø) IP                                                    MAIN 32Ø
      READ(LIN,1Ø) N                                                     MAIN 33Ø
   1Ø FORMAT(I2)                                                         MAIN 34Ø
      READ(LIN,2Ø) (YI(I),I=1,N)                                         MAIN 35Ø
      READ(LIN,2Ø) TI, TF, HI                                            MAIN 36Ø
      DO 1ØØØ K=1,NC                                                     MAIN 37Ø
      READ(LIN,1Ø) MF                                                    MAIN 38Ø
      READ(LIN,2Ø) EPS                                                   MAIN 39Ø
   2Ø FORMAT(3D22.15)                                                    MAIN 4ØØ
      WRITE(LOUT,3Ø) NP, MF, EPS                                         MAIN 41Ø
   3Ø FORMAT(1H1,1ØX,13H PROBLEM NO. ,I2//                               MAIN 42Ø
     1           11X,16H METHOD FLAG =  ,I2//                            MAIN 43Ø
     2           11X,18H ERROR PER STEP = ,D1Ø.3//                       MAIN 44Ø
     3           5X,4HTIME,15X,29HS O L U T I O N  /  E R R O R,26X,     MAIN 45Ø
     4           7H H-USED,3X,7HNQ-USED,3X,6H STEPS,3X,5H FNS ,3X,       MAIN 46Ø
     5           5H JACS//)                                              MAIN 47Ø
      DO 4Ø I=1,N                                                        MAIN 48Ø
C*                                                                       *MAIN 49Ø
C*       SET THE NORMALIZING VECTOR YMAX                                 *MAIN 5ØØ
C*                                                                       *MAIN 51Ø
      YMAX(I)=DMAX1(DABS(YI(I)),1.DØ)                                    MAIN 52Ø
C*                                                                       *MAIN 53Ø
      ERORI(I)=Ø.DØ                                                      MAIN 54Ø
   4Ø Y Ø(I,1,1)=YI(I)                                                   MAIN 55Ø
C*                                                                       *MAIN 56Ø
C*       THE FOLLOWING PARAMETERS ARE USED FOR STATISTICAL PURPOSES      *MAIN 57Ø
C*                                                                       *MAIN 58Ø
C*       NSTEP   NUMBER OF ACCEPTED STEPS                                *MAIN 59Ø
C*       NFE     NUMBER OF FUNCTION EVALUATIONS                          *MAIN 6ØØ
C*       NJE     NUMBER OF JACOBIAN EVALUATIONS                          *MAIN 61Ø
C*       NINVS   NUMBER OF CONVERGENCE FACTOR INVERSIONS                 *MAIN 62Ø
C*                                                                       *MAIN 63Ø
      NSTEP=Ø                                                            MAIN 64Ø
      NFE=Ø                                                              MAIN 65Ø
      NJE=Ø                                                              MAIN 66Ø
      NINVS=Ø                                                            MAIN 67Ø
C*                                                                       *MAIN 68Ø
C*       THE FIRST POINT AT WHICH THE SOLUTION IS SOUGHT IS SET BELOW.   *MAIN 69Ø
C*       FOR TESTING PURPOSES THE SET OF POINTS TOUT AT WHICH THE        *MAIN 7ØØ
C*       SOLUTION IS SOUGHT IS DESIGNED SO THAT ITS DENSITY IS LARGE     *MAIN 71Ø
C*       AT THE BEGINNING OF THE INTEGRATION INTERVAL AND THEN           *MAIN 72Ø
C*       GEOMETRICALLY DECREASES. TOUTP IS THE LAST INTERPOLATION POINT*MAIN 73Ø
C*       HMAX, THE MAXIMUM STEP-SIZE, IS EQUAL TO THE DISTANCE           *MAIN 74Ø
C*       BETWEEN THE NEXT AND LAST INTERPOLATION POINTS, MULTIPLIED      *MAIN 75Ø
C*       BY 1Ø.                                                          *MAIN 76Ø
C*                                                                       *MAIN 77Ø
      TOUT=TI+(TF-TI)/2.DØ**IP                                           MAIN 78Ø
      TOUTP=TI                                                           MAIN 79Ø
      HMAX=(TOUT-TOUTP)*1.D1                                             MAIN 8ØØ
C*                                                                       *MAIN 81Ø
      T =TI                                                              MAIN 82Ø
      HNEXT=HI                                                           MAIN 83Ø
      HMIN=HI*1.D-2                                                      MAIN 84Ø
      MAXDER=7                                                           MAIN 85Ø
      JSTART=Ø                                                           MAIN 86Ø
```

```
   5Ø CALL STINT (N, T, YØ, YDOT, SAVE, HØ, HNEXT, HMIN, HMAX, EPS,     MAIN 87Ø
     1           YMAX, KFLAG, KNEXT, JSTART, MAXDER, RW, RJ, MF, IPIV) MAIN 88Ø
      IF(KFLAG.GE.Ø) GO TO 2ØØ                                          MAIN 89Ø
      WRITE(LOUT,6Ø) KFLAG                                              MAIN 9ØØ
   6Ø FORMAT(1HØ/2ØH COMPLETION CODE IS ,I4)                            MAIN 91Ø
      GO TO 8ØØ                                                         MAIN 92Ø
  2ØØ KFLGP1=KFLAG+1                                                    MAIN 93Ø
C*                                                                      *MAIN 94Ø
C*       CHECK WHETHER THE COMPUTED SOLUTION REACHED BEYOND THE         *MAIN 95Ø
C*       INTERPOLATION POINT TOUT                                       *MAIN 96Ø
C*                                                                      *MAIN 97Ø
      II=Ø                                                              MAIN 98Ø
      DO 5ØØ I=1,KFLGP1                                                 MAIN 99Ø
      II=II+1                                                           MAIN1ØØØ
      TS=TOUT-T+HØ*DBLE(FLOAT(I-1))                                     MAIN1Ø1Ø
      IF(TS.LT.Ø.DØ) GO TO 5ØØ                                          MAIN1Ø2Ø
      IF(I-1) 5Ø, 5Ø, 51Ø                                               MAIN1Ø3Ø
  5ØØ CONTINUE                                                          MAIN1Ø4Ø
C*                                                                      *MAIN1Ø5Ø
C*       THE SOLUTION REACHED BEYOND TOUT.                              *MAIN1Ø6Ø
C*       PERFORM INTERPOLATION AT TOUT.                                 *MAIN1Ø7Ø
C*                                                                      *MAIN1Ø8Ø
  51Ø IND=KFLAG+3-II                                                    MAIN1Ø9Ø
      IF(II.EQ.2) IND=1                                                 MAIN11ØØ
      S=(TS-HØ)/HØ                                                      MAIN111Ø
      DO 6ØØ I=1,N                                                      MAIN112Ø
      D=1.DØ                                                            MAIN113Ø
      YOUT(I)=YØ(I,1,IND)                                               MAIN114Ø
      DO 6ØØ J=1,JSTART                                                 MAIN115Ø
      D=D*(DBLE(FLOAT(J-1))+S)/DBLE(FLOAT(J))                           MAIN116Ø
  6ØØ YOUT(I)=YOUT(I)+D*YØ(I,J+1,IND)                                   MAIN117Ø
      CALL ERROR(TOUT,YOUT,N,ERO)                                       MAIN118Ø
C*                                                                      *MAIN119Ø
C*       DETERMINE THE MAXIMUM ERROR OF THE INTERPOLATED SOLUTION       *MAIN12ØØ
C*       COMMITTED SO FAR AND PRINT IT OUT WITH THE SOLUTION AT TOUT    *MAIN121Ø
C*                                                                      *MAIN122Ø
      DO 61Ø I=1,N                                                      MAIN123Ø
  61Ø ERORI(I)=DMAX1(ERORI(I),DABS(ERO(I)))                             MAIN124Ø
      WRITE(LOUT,62Ø) TOUT, (YOUT(I),I=1,N), HØ, JSTART, NSTEP, NFE,    MAIN125Ø
     1                NJE, (ERO(I),I=1,N)                              MAIN126Ø
  62Ø FORMAT( 4H T =,D11.4,4D15.6,D12.3,3X,I4,6X,I4,4X,I4,4X,I4/15X,    MAIN127Ø
     1        4D15.6/)                                                  MAIN128Ø
C*                                                                      *MAIN129Ø
C*       SET THE NEW INTERPOLATION POINT AND CHECK WHETHER AN           *MAIN13ØØ
C*       INTERPOLATION CAN BE PERFORMED.  SET NEW HMAX                  *MAIN131Ø
C*                                                                      *MAIN132Ø
      TOUTP=TOUT                                                        MAIN133Ø
      TOUT=TI+(TOUT-TI)*2.DØ                                            MAIN134Ø
      IF(TOUT.GT.TF) GO TO 8ØØ                                          MAIN135Ø
      IF(TOUT.LE.T) GO TO 2ØØ                                           MAIN136Ø
      HMAX=(TOUT-TOUTP)*1.D1                                            MAIN137Ø
      GO TO 5Ø                                                          MAIN138Ø
C*                                                                      *MAIN139Ø
C*       REGARDLESS OF WHETHER OR NOT THE PROBLEM HAS BEEN SOLVED,      *MAIN14ØØ
C*       PRINT OUT THE FINAL STATISTICS                                 *MAIN141Ø
C*                                                                      *MAIN142Ø
  8ØØ WRITE(LOUT,81Ø)                                                   MAIN143Ø
  81Ø FORMAT(1HØ/2ØX,2ØHINTERPOLATION ERRORS)                           MAIN144Ø
      WRITE(LOUT,82Ø) (ERORI(I),I=1,N)                                  MAIN145Ø
  82Ø FORMAT(1HØ/1ØX,23H MAXIMUM ABSOLUTE ERROR//11X,4D22.8)            MAIN146Ø
      DO 83Ø I=1,N                                                      MAIN147Ø
  83Ø ERORI(I)=ERORI(I)/YMAX(I)                                         MAIN148Ø
      WRITE(LOUT,84Ø) (ERORI(I),I=1,N)                                  MAIN149Ø
  84Ø FORMAT(1HØ/1ØX,23H MAXIMUM RELATIVE ERROR//11X,4D22.8)            MAIN15ØØ
      WRITE(LOUT,85Ø) NSTEP, NFE, NJE, NINVS                            MAIN151Ø
  85Ø FORMAT(1HØ/22H PROBLEM COMPLETED IN ,I5,6H STEPS/
     1           22X,I5,21H FUNCTION EVALUATIONS/                       MAIN153Ø
     2           22X,I5,21H JACOBIAN EVALUATIONS/                       MAIN154Ø
     3           22X,I5,18H LU DECOMPOSITIONS)                          MAIN155Ø
 1ØØØ CONTINUE                                                          MAIN156Ø
      STOP                                                              MAIN157Ø
      END                                                               MAIN158Ø
```

```
      SUBROUTINE STINT (N, T, Y, DY, SAVE, H, HNEXT, HMIN, HMAX, EPS,   STNT  1Ø
     +                  YMAX, KFLAG, KNEXT, JSTART, MAXORD, RW, RJ, MF, STNT  2Ø
     +                  IPIV)                                           STNT  3Ø
C                                                                       STNT  4Ø
C***********************************************************************STNT  5Ø
C*                                                                      *STNT  6Ø
C*    THIS PROGRAM INTEGRATES A SET OF N FIRST ORDER ORDINARY           *STNT  7Ø
C*   DIFFERENTIAL EQUATIONS.  A BLOCK OF THREE OR FOUR SOLUTION POINTS,*STNT  8Ø
C*   EACH SEPARATED BY A STEP-SIZE H, IS COMPUTED AT EACH CALL. THE     *STNT  9Ø
C*   STEP-SIZE MAGNITUDE MAY BE SPECIFIED BY THE USER AT EACH CALL.     *STNT 1ØØ
C*   ALTERNATIVELY, IT MAY BE INCREASED OR DECREASED BY STINT WITHIN    *STNT 11Ø
C*   THE RANGE DABS(HMIN) TO DABS(HMAX),                                *STNT 12Ø
C*   IN ORDER TO ACHIEVE AS LARGE A STEP AS POSSIBLE, WHILE NOT         *STNT 13Ø
C*   COMMITING A SINGLE STEP ERROR WHICH IS LARGER THAN EPS IN THE      *STNT 14Ø
C*   RMS NORM, WHERE EACH COMPONENT OF THE ERROR VECTOR IS DIVIDED BY   *STNT 15Ø
C*   THE CORRESPONDING COMPONENT OF YMAX.                               *STNT 16Ø
C*                                                                      *STNT 17Ø
C*    THE PROGRAM REQUIRES 4 SUBROUTINES NAMED                          *STNT 18Ø
C*                                                                      *STNT 19Ø
C*         DIFFUN(N, T, Y, DY)                                          *STNT 2ØØ
C*         PEDERV(N, T, Y, RJ, NØ)                                      *STNT 21Ø
C*         DEC(N, NØ, A, IPIV, IER)                                     *STNT 22Ø
C*         SOL(N, NØ, A, B, IPIV)                                       *STNT 23Ø
C*                                                                      *STNT 24Ø
C*    DIFFUN EVALUATES THE DERIVATIVES OF THE DEPENDENT VARIABLE Y(I)   *STNT 25Ø
C*   FOR I=1,...,N AND T AND STORES THE RESULT IN THE ARRAY DY.         *STNT 26Ø
C*                                                                      *STNT 27Ø
C*    PEDERV COMPUTES THE PARTIAL DERIVATIVES OF THE DIFFERENTIAL       *STNT 28Ø
C*   EQUATIONS AT THE VALUES Y(I) FOR I=1,...,N AND T AND STORES THE    *STNT 29Ø
C*   RESULT IN ARRAY RJ.  THUS, RJ(J+(K-1)*NØ) IS THE PARTIAL OF DY(J)  *STNT 3ØØ
C*   WITH RESPECT TO Y(K) FOR J,K=1,...,N EVALUATED AT Y(I) FOR I=1,..  *STNT 31Ø
C*   ..,N AND T.  NOTE--NØ IS THE VALUE OF N ON THE FIRST CALL.  IF THE*STNT 32Ø
C*   ANALYTIC EXPRESSIONS FOR THE PARTIAL DERIVATIVES ARE NOT           *STNT 33Ø
C*   AVAILABLE, THEIR APPROXIMATE VALUES CAN BE OBTAINED BY NUMERICAL   *STNT 34Ø
C*   DIFFERENCING.  (SEE PARAMETER MF.)  IN THIS CASE SUBROUTINE PEDERV*STNT 35Ø
C*   MAY SIMPLY BE                                                      *STNT 36Ø
C*                                                                      *STNT 37Ø
C*         SUBROUTINE PEDERV(N, T, Y, RJ, NØ)                           *STNT 38Ø
C*         RETURN                                                       *STNT 39Ø
C*         END                                                          *STNT 4ØØ
C*                                                                      *STNT 41Ø
C*    DEC PERFORMS AN LU DECOMPOSITION OF A MATRIX A. IF THE            *STNT 42Ø
C*   DECOMPOSITION IS SUCCESSFUL IER SHOULD BE SET TO Ø, OTHERWISE IT   *STNT 43Ø
C*   SHOULD BE SET TO +1.                                               *STNT 44Ø
C*                                                                      *STNT 45Ø
C*    SOL SOLVES THE LINEAR ALGEBRAIC SYSTEM A*X=B, FOR WHICH THE       *STNT 46Ø
C*   MATRIX A WAS PROCESSED BY DEC.                                     *STNT 47Ø
C*                                                                      *STNT 48Ø
C*    THIS PROGRAM USES DOUBLE PRECISION FOR ALL FLOATING POINT         *STNT 49Ø
C*   VARIABLES, EXCEPT FOR THOSE STARTING WITH P, WHICH ARE IN SINGLE   *STNT 5ØØ
C*   PRECISION.                                                         *STNT 51Ø
C*                                                                      *STNT 52Ø
C*    TEMPORARY STORAGE SPACE IS PROVIDED (BY THE USER) IN THE          *STNT 53Ø
C*   ARRAYS IPIV, RJ, RW, AND SAVE. THE ARRAY IPIV HOLDS A VECTOR OF    *STNT 54Ø
C*   THE SAME NAME. THE ARRAYS RJ AND RW ARE USED TO HOLD MATRICES OF   *STNT 55Ø
C*   THE SAME NAMES.  THE ARRAY SAVE IS PARTITIONED AS FOLLOWS          *STNT 56Ø
C*                                                                      *STNT 57Ø
C*    SAVE(J,I)      1.LE.J.LE.8 AND 1.LE.I.LE.N IS USED TO SAVE       *STNT 58Ø
C*                   Y(I,J) IN CASE A STEP (AND HENCE THE WHOLE CYCLE)  *STNT 59Ø
C*                   HAS TO BE REPEATED.                                *STNT 6ØØ
C*    SAVE(9,I)      1.LE.I.LE.N IS USED TO STORE THE DERIVATIVE OF THE *STNT 61Ø
C*                   I-TH DEPENDENT VARIABLE SCALED BY H.               *STNT 62Ø
C*    SAVE(N9+I,1)   IS USED TO STORE THE DERIVATIVES AS THEY ARE       *STNT 63Ø
C*                   COMPUTED BY DIFFUN FOR THE CORRECTOR. IT IS ALSO   *STNT 64Ø
C*                   ACCESSED AS A COMPLETE ARRAY SAVE(N9P1,1).         *STNT 65Ø
C*                   IN ADDITION IT IS USED IN THE ERRCR CONTROL TEST.  *STNT 66Ø
C*                   (N9 = NØ * 9  AND  N9P1 = N9 + 1.)                 *STNT 67Ø
C*    SAVE(N1Ø+I,1) IS USED TO HOLD THE CORRECTION TERMS FOR THE ENTIRE*STNT 68Ø
C*                   CORRECTOR ITERATION IN THE CASE, THE CORRECTOR HAS *STNT 69Ø
C*                   TO BE REPEATED. (N1Ø = NØ * 1Ø.)                   *STNT 7ØØ
C*    SAVE(N11+I,1) IS USED TO HOLD THE DERIVATIVES EVALUATED AT        *STNT 71Ø
C*                   Y(I)+D AND T, WHERE D IS THE INCREMENT USED IN THE *STNT 72Ø
C*                   NUMERICAL DIFFERENCING SCHEME INVOKED, IN ORDER TO *STNT 73Ø
C*                   OBTAIN APPROXIMATE VALUES OF THE PARTIAL           *STNT 74Ø
C*                   DERIVATIVES. (N11 = NØ * 11.)                      *STNT 75Ø
C*    SAVE(N12+I,1) IS USED TO HOLD THE DERIVATIVES EVALUATED AT Y(I)   *STNT 76Ø
```

```
C*                AND T IN ORDER TO OBTAIN APPROXIMATE VALUES OF THE *STNT 77Ø
C*                PARTIAL DERIVATIVES. (N12 = NØ * 12.)              *STNT 78Ø
C*                                                                   *STNT 79Ø
C*   THE PARAMETERS IN THE CALLING SEQUENCE HAVE THE FOLLOWING MEANING *STNT 8ØØ
C*                                                                   *STNT 81Ø
C*    N       THE NUMBER OF FIRST ORDER DIFFERENTIAL EQUATIONS TO BE  *STNT 82Ø
C*             INTEGRATED.  N MAY BE DECREASED ON SUBSEQUENT CALLS IF *STNT 83Ø
C*             THE NUMBER OF ACTIVE EQUATIONS DECREASES, WITH THE     *STNT 84Ø
C*             FIRST ONES BEING THO$E RETAINED.  BUT, IT MUST         *STNT 85Ø
C*             NOT BE INCREASED WITHOUT CALLING WITH JSTART = Ø.       *STNT 86Ø
C*    T       THE INDEPENDENT VARIABLE.  ON ENTRY T IS THE CURRENT    *STNT 87Ø
C*             SETTING OF THE INDEPENDENT VARIABLE.  ON RETURN TO THE *STNT 88Ø
C*             CALLING PROGRAM, T CORRESPONDS TO THE SETTING OF THE   *STNT 89Ø
C*             INDEPENDENT VARIABLE FOR THE MOST FORWARD POINT OBTAINED*STNT 9ØØ
C*             THUS FAR.                                             *STNT 91Ø
C*    Y       AN N BY 8 BY 4 ARRAY CONTAINING THE DEPENDENT VARIABLES *STNT 92Ø
C*             AND THEIR BACKWARD DIFFERENCES.  ON EACH CALL UP TO FOUR*STNT 93Ø
C*             SOLUTION POINTS ARE OBTAINED. THE MOST FORWARD POINT IS *STNT 94Ø
C*             ALWAYS AT Y(I,1,1). THE POINT NEXT TO THE MOST FORWARD  *STNT 95Ø
C*             POINT IS RETURNED IN Y(I,1,KFLAG). THE MOST BACKWARD    *STNT 96Ø
C*             POINT IN THE NEW BLOCK IS RETURNED IN Y(I,1,2). ONLY THE*STNT 97Ø
C*             INITIAL SOLUTION VALUES, ENTERED IN Y(I,1,1) FOR        *STNT 98Ø
C*             I=1,...,N, NEED TO BE SUPPLIED ON THE FIRST             *STNT 99Ø
C*             CALL (JSTART = Ø).                                     *STNT1ØØØ
C*             Y(I,J+1,K) CONTAINS THE J-TH BACKWARD DIFFERENCE OF THE *STNT1Ø1Ø
C*             I-TH DEPENDENT VARIABLE (FOR K=1,...,KFLAG).           *STNT1Ø2Ø
C*             IF IT IS DESIRED TO INTERPOLATE TO NON-MESH POINTS,     *STNT1Ø3Ø
C*             THESE VALUES CAN BE USED.  IF THE CURRENT STEP-SIZE IS  *STNT1Ø4Ø
C*             H AND THE VALUE AT T-E (Ø.LT.DABS(E).LT.DABS(H)) IS     *STNT1Ø5Ø
C*             NEEDED, FORM S=E/H AND THEN COMPUTE                     *STNT1Ø6Ø
C*                              NQ                                     *STNT1Ø7Ø
C*                 Y(I)(T-E) = SUM Y(I,J+1,K)*B(J)                     *STNT1Ø8Ø
C*                             J=Ø                                     *STNT1Ø9Ø
C*             WHERE K, WHICH CORRESPONDS TO T, IS THE FIRST MESH POINT*STNT11ØØ
C*             BEYOND THE POINT T-E, AND B(J+1) = B(J)*(J-S)/(J+1)     *STNT1111Ø
C*             WITH B(Ø) = 1.                                         *STNT1112Ø
C*    DY      AN N BY 4 ARRAY.  DY(I,K), 1.LE.I.LE.N, 1.LE.K.LE.KFLAG, *STNT1113Ø
C*             CONTAINS THE DERIVATIVE OF THE I-TH DEPENDENT VARIABLE  *STNT1114Ø
C*             SCALED BY H.  THE DY(I,1) ARRAY NEED NOT BE SUPPLIED AT *STNT1115Ø
C*             THE FIRST CALL (JSTART = Ø).                           *STNT1116Ø
C*    SAVE    A BLOCK OF AT LEAST 13*NØ DOUBLE PRECISION FLOATING POINT*STNT1117Ø
C*             LOCATIONS.                                             *STNT1118Ø
C*    H       THE STEP-SIZE USED FOR THE JUST COMPLETED BLOCK.        *STNT1119Ø
C*    HNEXT   THE STEP-SIZE FOR THE NEXT BLOCK.  ON THE               *STNT12ØØ
C*             FIRST CALL (JSTART=Ø) THE USER MUST SPECIFY AN INITIAL  *STNT121Ø
C*             STEP-SIZE.  A GOOD ESTIMATE OF ITS MAGNITUDE IS GIVEN   *STNT122Ø
C*             BY  Ø.2*(EPS/DABS(E))**Ø.5, WHERE E IS THE LARGEST      *STNT123Ø
C*             EIGENVALUE OF THE JACOBIAN EVALUATED AT THE INITIAL     *STNT124Ø
C*             VALUES OF T AND Y.  THE SIGN OF THE INITIAL HNEXT       *STNT125Ø
C*             SHOULD BE POSITIVE (NEGATIVE) IF THE FINAL TIME IS      *STNT126Ø
C*             GREATER (LESS) THAN THE INITIAL TIME.  IF THE INITIAL   *STNT127Ø
C*             STEP-SIZE CHOICE DOES NOT CAUSE AN ERROR GREATER THAN   *STNT128Ø
C*             EPS, IN THE RMS NORM, IT WILL BE ACCEPTED. OTHERWISE,   *STNT129Ø
C*             ITS MAGNITUDE WILL BE DECREASED UNTIL AN ERROR LESS     *STNT13ØØ
C*             THAN EPS IS ACHIEVED.  THE SUBROUTINE AUTOMATICALLY     *STNT131Ø
C*             ADJUSTS THE STEP-SIZE AFTER THE INITIAL AND SUBSEQUENT  *STNT132Ø
C*             CALLS FOR THE STEP-SIZE OF LARGEST POSSIBLE MAGNITUDE.  *STNT133Ø
C*             THE MAGNITUDE MAY BE ADJUSTED DOWN ON ANY SUBSEQUENT    *STNT134Ø
C*             CALL.  NOTE--A MAGNITUDE ADJUSTMENT UP OR A SIGN CHANGE *STNT135Ø
C*             WILL BE IGNORED.                                        *STNT136Ø
C*    HMIN    DABS(HMIN) IS THE MINIMUM STEP-SIZE MAGNITUDE TO BE     *STNT137Ø
C*             ALLOWED FOR THE NEXT INTEGRATION CYCLE.  (ON THE FIRST  *STNT138Ø
C*             CALL (JSTART=Ø), DABS(HMIN) SHOULD BE CHOSEN SIGNIF-    *STNT139Ø
C*             ICANTLY SMALLER THAN DABS(HNEXT) SO AS TO AVOID START-UP*STNT14ØØ
C*             PROBLEMS IF THE ERROR CRITERION IS NOT MET WITH THE USER*STNT141Ø
C*             SPECIFIED HNEXT.)  NOTE--THE SIGN OF HMIN IS IGNORED.   *STNT142Ø
C*             HMIN MAY BE CHANGED ON SUBSEQUENT CALLS.                *STNT143Ø
C*    HMAX    DABS(HMAX) IS THE MAXIMUM STEP-SIZE MAGNITUDE TO BE     *STNT144Ø
C*             ALLOWED FOR THE NEXT INTEGRATION CYCLE.  NOTE--IF       *STNT145Ø
C*             DABS(HMAX) IS LESS THAN DABS(HMIN), THEN THE SUBROUTINE *STNT146Ø
C*             FUNCTIONS AS THOUGH DABS(HMAX) EQUALS DABS(HMIN). HMAX  *STNT147Ø
C*             MAY BE CHANGED ON SUBSEQUENT CALLS.                     *STNT148Ø
C*    EPS     THE ERROR TEST CONSTANT.  THE SINGLE STEP ERROR ESTIMATE *STNT149Ø
C*             FOR Y, COMPUTED AS A WEIGHTED RMS NORM, MUST NOT EXCEED *STNT15ØØ
C*             EPS.  THE WEIGHT FOR THE I-TH ELEMENT IN THE ERROR      *STNT151Ø
C*             ESTIMATE IS 1/YMAX(I).  (SEE PARAMETER YMAX.)  THE      *STNT152Ø
C*             STEP-SIZE AND/OR ORDER ARE ADJUSTED TO ACHIEVE THIS.    *STNT153Ø
```

```
C*    YMAX    AN ARRAY OF NØ LOCATIONS, WITH--FOR I=1,...,N--YMAX(I)   *STNT154Ø
C*             BEING THE MAXIMUM OF UNITY AND THE MAXIMUM VALUE OF     *STNT155Ø
C*             DABS(Y(I)) SEEN THUS FAR.  ON THE FIRST CALL IT SHOULD  *STNT156Ø
C*             BE SET TO THE MAXIMUM OF UNITY AND THE INITIAL VALUE OF *STNT157Ø
C*             DABS(Y(I)).                                             *STNT158Ø
C*    KFLAG   A COMPLETION CODE. IF KFLAG IS GREATER THAN Ø,THEN       *STNT159Ø
C*             KFLAG POINTS HAVE BEEN COMPUTED. IF KFLAG IS            *STNT16ØØ
C*             -2, -3, OR -4 THEN 2, 3, OR 4 MESH POINTS, RESPECTIVELY,*STNT161Ø
C*             HAVE BEEN COMPUTED WITH DABS(H) EQUAL TO DABS(HMIN),    *STNT162Ø
C*             BUT THE REQUESTED ERROR WAS NOT ACHIEVED.  OTHER VALUES *STNT163Ø
C*             KFLAG CAN ASSUME ARE AS FOLLOWS                         *STNT164Ø
C*              -5  THE REQUESTED ERROR WAS SMALLER THAN CAN BE HANDLED*STNT165Ø
C*                  FOR THIS PROBLEM.                                  *STNT166Ø
C*              -6  CORRECTOR CONVERGENCE COULD NOT BE ACHIEVED FOR    *STNT167Ø
C*                  DABS(H).GT.DABS(HMIN).                             *STNT168Ø
C*              -7  THE MAXIMUM ORDER SPECIFIED WAS TOO LARGE.         *STNT169Ø
C*    KNEXT   AFTER THE INITIAL CALL (JSTART=Ø), THE VALUE OF KNEXT IS *STNT17ØØ
C*             THE NUMBER OF POINTS TO BE COMPUTED DURING THE NEXT     *STNT171Ø
C*             CYCLE.  THE VALUE IS SUPPLIED FOR INFORMATION ONLY.  THE*STNT172Ø
C*             USER CANNOT CONTROL THE NUMBER OF POINTS BY SETTING     *STNT173Ø
C*             KNEXT.                                                  *STNT174Ø
C*    JSTART  AN INPUT INDICATOR WITH THE FOLLOWING MEANINGS           *STNT175Ø
C*              .EQ.Ø  INITIALIZATION CALL.  (JSTART MUST BE SET TO Ø  *STNT176Ø
C*                     ON THE FIRST CALL.)                             *STNT177Ø
C*              .GT.Ø  CONTINUE FROM THE LAST STEP.                    *STNT178Ø
C*             ON RETURN JSTART IS SET TO NQ, THE MAXIMUM BACKWARD     *STNT179Ø
C*             DIFFERENCE AVAILABLE IN THE Y ARRAY.  (THIS ALSO CORRES-*STNT18ØØ
C*             PONDS TO THE ORDER OF THE METHOD USED TO COMPUTE THE    *STNT181Ø
C*             JUST COMPLETED BLOCK OF POINTS.)                        *STNT182Ø
C*    MAXORD  THE MAXIMUM ORDER (1.LE.MAXORD.LE.7) THAT MAY BE USED.   *STNT183Ø
C*             NOTE--IF MAXORD IS RESET BETWEEN CYCLES TO A VALUE LESS *STNT184Ø
C*             THAN THE ORDER DETERMINED FOR THE NEXT CYCLE, THEN THE  *STNT185Ø
C*             ORDER MAY FOR SEVERAL CYCLES EXCEED MAXORD.  HOWEVER,   *STNT186Ø
C*             IT CANNOT EXCEED THE ABOVE DETERMINED VALUE AND, ONCE   *STNT187Ø
C*             THE ORDER IS LESS THAN OR EQUAL TO MAXORD, IT CANNOT    *STNT188Ø
C*             THEN EXCEED MAXORD.                                     *STNT189Ø
C*    RJ      A BLOCK OF AT LEAST NØ**2 DOUBLE PRECISION FLOATING POINT*STNT19ØØ
C*             LOCATIONS, WHICH CONTAINS AN ESTIMATE OF THE JACOBIAN   *STNT191Ø
C*             OF THE DIFFERENTIAL EQUATION.                           *STNT192Ø
C*    RW      A BLOCK OF AT LEAST NØ**2 DOUBLE PRECISION FLOATING POINT*STNT193Ø
C*             LOCATIONS.                                              *STNT194Ø
C*    IPIV    A BLOCK OF AT LEAST NØ INTEGERS USED TO HOLD PIVOT DATA  *STNT195Ø
C*             GENERATED DURING AN LU DECOMPOSITION.                   *STNT196Ø
C*    MF      METHOD FLAG.  IT DETERMINES THE MODE BY WHICH THE PARTIAL*STNT197Ø
C*             DERIVATIVES ARE OBTAINED WITH THE FOLLOWING MEANINGS    *STNT198Ø
C*              1  ANALYTIC EXPRESSIONS FOR THE PARTIAL DERIVATIVES ARE *STNT199Ø
C*                 SUPPLIED BY THE USER IN THE SUBROUTINE PEDERV.       *STNT2ØØØ
C*              2  THE ANALYTIC EXPRESSIONS ARE NOT AVAILABLE.          *STNT2Ø1Ø
C*                 APPROXIMATE VALUES OF THE PARTIAL DERIVATIVES ARE    *STNT2Ø2Ø
C*                 OBTAINED BY NUMERICAL DIFFERENCING.                  *STNT2Ø3Ø
C*                                                                      *STNT2Ø4Ø
C***********************************************************************STNT2Ø5Ø
C                                                                        STNT2Ø6Ø
      DOUBLE PRECISION B, BND, B1, B2, C, CRATE, D, DF, DI, DY,          STNT2Ø7Ø
     +                 D1, D2, D3, E, EDOWN, ENQDWN, ENQSAM,             STNT2Ø8Ø
     +                 ENQUP, EPS, ES, EUP, FN, H, HMAX, HMIN,           STNT2Ø9Ø
     +                 HNEW, HNEXT, HOLD, Q, RATIO, RC, RJ, RMAX,        STNT21ØØ
     +                 RRDOWN, RRSAME, RRUP, RW, SAVE, SQRTUR, T,        STNT211Ø
     +                 TDL, TOLD, UROUND, Y, YJ1, YMAX                   STNT212Ø
C     DOUBLE PRECISION DABS, DBLE, DMAX1, DMIN1, DSQRT                   STNT213Ø
      DIMENSION Y(N,8,4), DY(N,4), SAVE(9,1), YMAX(1), RW(1), IPIV(1),   STNT214Ø
     +          INDEX(7,2), B(82), C(16), PERROR(9), PC(16), PD(7),      STNT215Ø
     +          B1(48), B2(34), RJ(1)                                    STNT216Ø
      COMMON /STAT/ NSTEP,NFE,NJE,NINVS                                  STNT217Ø
      EQUIVALENCE(B(1),B1(1)),(B(49),B2(1))                              STNT218Ø
C                                                                        STNT219Ø
C         ***************************************************            STNT22ØØ
C         *****  SEVEN OF EIGHT DATA STATEMENTS CONTAIN  *****           STNT221Ø
C         *****  ARRAY NAMES AND VARIOUS  LINES OF CODE  *****           STNT222Ø
C         *****  INCLUDE SUBSCRIPT EXPRESSIONS.          *****           STNT223Ø
C         ***************************************************            STNT224Ø
C                                                                        STNT225Ø
C***********************************************************************STNT226Ø
C*   THE CONSTANT UROUND SHOULD BE SET EQUAL TO THE UNIT ROUND-OFF      *STNT227Ø
C*   FOR THE MACHINE.  THE CONSTANT SQRTUR SHOULD BE SET EQUAL TO THE   *STNT228Ø
C*   SQUARE ROOT OF UROUND.                                             *STNT229Ø
C***********************************************************************STNT23ØØ
```

```
C                                                                       STNT231Ø
      DATA UROUND/4.D-16/, SQRTUR/2.D-8/                                STNT232Ø
C                                                                       STNT233Ø
C***********************************************************************STNT234Ø
C*   THE ARRAY INDEX HOLDS POINTERS AND CONSTANTS FOR THE VARIOUS       *STNT235Ø
C*   ORDER METHODS.  FOR NQ=1,...,7 THE ENTRIES ARE AS FOLLOWS          *STNT236Ø
C*    INDEX(NQ,1)   BASE INDEX FOR B ARRAY (H*DY PREDICTOR).            *STNT237Ø
C*    INDEX(NQ,2)   BASE INDEX FOR C ARRAY (CORRECTOR).                 *STNT238Ø
C***********************************************************************STNT239Ø
C                                                                       STNT24ØØ
      DATA INDEX/ 1, 2, 4, 11, 2Ø, 38, 59,                             STNT241Ø
     +            1, 2, 3,  5,  7, 1Ø, 14/                            STNT242Ø
C                                                                       STNT243Ø
C***********************************************************************STNT244Ø
C*   THE COEFFICIENTS APPEARING IN THE NEXT DATA STATEMENTS FOR THE     *STNT245Ø
C*   B ARRAY SHOULD BE DEFINED TO THE MAXIMUM ACCURACY PERMITTED BY     *STNT246Ø
C*   THE MACHINE. THEY ARE, IN THE ORDER SPECIFIED,...                  *STNT247Ø
C*    1                                                                 *STNT248Ø
C*    -2, 3                                                             *STNT249Ø
C*    -9/2, -5/4, 11/2                                                  *STNT25ØØ
C*    -15/2, -3/4, 13/2, 2                                              *STNT251Ø
C*    -22/3, -8/3, -17/18, 25/3                                         *STNT252Ø
C*    -9, -9/4, -7/8, 35/4, 5/4                                         *STNT253Ø
C*    -125/12, -1Ø1/24, -71/36, -37/48, 137/12                          *STNT254Ø
C*    -253/24, -1ØØ1/24Ø, -7Ø7/36Ø, -123/16Ø, 1373/12Ø, 1/1Ø            *STNT255Ø
C*    -57/4, -25/8, -17/1Ø, -169/24Ø, 61/5, -1/2Ø, 31/1Ø                *STNT256Ø
C*    -137/1Ø, -117/2Ø, -46/15, -191/12Ø, -197/3ØØ, 147/1Ø              *STNT257Ø
C*    -353/25, -571/1ØØ, -1819/6ØØ, -79/5Ø, -3919/6ØØØ, 1477/1ØØ, 7/2Ø  *STNT258Ø
C*    -3529/2ØØ, -1889/4ØØ, -16Ø9/6ØØ, -3527/24ØØ, -931/15ØØ, 3Ø79/2ØØ,*STNT259Ø
C*        -3/2Ø, 17/5                                                   *STNT26ØØ
C*    -343/2Ø, -3Ø3/4Ø, -253/6Ø, -589/24Ø, -1Ø1/75, -23/4Ø, 363/2Ø      *STNT261Ø
C*    -266/15, -221/3Ø, -749/18Ø, -73/3Ø, -8Ø3/6ØØ, -1Ø3/18Ø, 547/3Ø,  *STNT262Ø
C*        1/2                                                           *STNT263Ø
C*    -1316/75, -1151/15Ø, -3689/9ØØ, -121/5Ø, -4ØØ3/3ØØØ, -257/45Ø,   *STNT264Ø
C*        2737/15Ø, -1/5, 1/2                                           *STNT265Ø
C***********************************************************************STNT266Ø
C                                                                       STNT267Ø
      DATA B1/                                                          STNT268Ø
     +        1.ØDØ, -2.ØDØ, 3.ØDØ, -4.5DØ, -1.25DØ, 5.5DØ, -7.5DØ,    STNT269Ø
     +        -.75DØ, 6.5DØ, 2.ØDØ, -7.3333333333333333DØ,              STNT27ØØ
     +        -2.6666666666666667DØ, -.94444444444444444DØ,             STNT271Ø
     +        8.3333333333333333DØ, -9.ØDØ, -2.25DØ, -.875DØ, 8.75DØ,  STNT272Ø
     +        1.25DØ, -1Ø.416666666666667DØ, -4.2Ø833333333333333DØ,    STNT273Ø
     +        -1.9722222222222222DØ, -.77Ø83333333333333DØ,             STNT274Ø
     +        11.416666666666667DØ, -1Ø.541666666666667DØ,              STNT275Ø
     +        -4.17Ø8333333333333DØ, -1.9638888888888889DØ, -.76875DØ,  STNT276Ø
     +        11.441666666666667DØ, Ø.1DØ, -14.25DØ, -3.125DØ, -1.7DØ,  STNT277Ø
     +        -.7Ø4166666666666667DØ, 12.2DØ, -.Ø5DØ, 3.1DØ, -13.7DØ,   STNT278Ø
     +        -5.85DØ, -3.Ø666666666666667DØ, -1.5916666666666667DØ,    STNT279Ø
     +        -.65666666666666667DØ, 14.7DØ, -14.12DØ, -5.71DØ,         STNT28ØØ
     +        -3.Ø3166666666666667DØ, -1.58DØ, -.65316666666666667DØ/   STNT281Ø
      DATA B2/                                                          STNT282Ø
     +        14.77DØ, Ø.35DØ, -17.645DØ, -4.7225DØ,                    STNT283Ø
     +        -2.6816666666666667DØ, -1.4695833333333333DØ,             STNT284Ø
     +        -.62Ø6666666666666667DØ, 15.395DØ, -.15DØ, 3.4DØ, -17.15DØ, STNT285Ø
     +        -7.575DØ, -4.2166666666666667DØ, -2.4541666666666667DØ,   STNT286Ø
     +        -1.3466666666666667DØ, -.575DØ, 18.15DØ,                  STNT287Ø
     +        -17.733333333333333DØ, -7.3666666666666667DØ,             STNT288Ø
     +        -4.1611111111111111DØ, -2.4333333333333333DØ,             STNT289Ø
     +        -1.3383333333333333DØ, -.57222222222222222DØ,             STNT29ØØ
     +        18.233333333333333DØ, Ø.5DØ, -17.546666666666667DØ,       STNT291Ø
     +        -7.6733333333333333DØ, -4.Ø988888888888889DØ, -2.42DØ,    STNT292Ø
     +        -1.3343333333333333DØ, -.57111111111111111DØ,             STNT293Ø
     +        18.246666666666667DØ, -Ø.2DØ, Ø.5DØ/                      STNT294Ø
C                                                                       STNT295Ø
C***********************************************************************STNT296Ø
C*   THE COEFFICIENTS APPEARING IN THE NEXT DATA STATEMENTS FOR THE     *STNT297Ø
C*   C ARRAY SHOULD BE DEFINED TO THE MAXIMUM ACCURACY PERMITTED BY     *STNT298Ø
C*   THE MACHINE.  THEY ARE, IN THE ORDER SPECIFIED,...                 *STNT299Ø
C*    -1                                                                *STNT3ØØØ
C*    -2/3                                                              *STNT3Ø1Ø
C*    -6/11, -2/3                                                       *STNT3Ø2Ø
C*    -12/25, -16/31                                                    *STNT3Ø3Ø
C*    -6Ø/137, -6ØØ/1373, -1ØØ/193                                      *STNT3Ø4Ø
C*    -2Ø/49, -12Ø/293, -75/184, -12ØØ/2299                             *STNT3Ø5Ø
C*    -14Ø/363, -6Ø/143, -1Ø5Ø/2437                                     *STNT3Ø6Ø
C***********************************************************************STNT3Ø7Ø
```

```
C
      DATA C/ -1.0D0, -.66666666666666667D0, -.54545454545454545D0,
     +        -.66666666666666667D0, -.48D0, -.51612903225806452D0,
     +        -.43795620437956204D0, -.43699927166788056D0,
     +        -.51813471502590674D0, -.40816326530612245D0,
     +        -.40955631399317406D0, -.40760869565217391D0,
     +        -.52196607220530666D0, -.38567493112947659D0,
     +        -.41958041958041958D0, -.43085761181780879D0/
C
C***********************************************************************
C*    THE COEFFICIENTS IN THE PERROR ARRAY ARE USED IN THE ERROR TEST, *
C*    THE FIRST TIME IT IS PERFORMED, AS WELL AS IN THE STEP-SIZE/ORDER *
C*    SELECTION SEGMENT. PERROR(I+1) = 1/D(I+1), I=1,...,7, WHERE      *
C*    D(I+1) IS THE DISCRETIZATION ERROR CONSTANT CORRESPONDING TO THE *
C*    SECOND PASS OF THE INTEGRATION CYCLE OF ORDER I. PERROR(1) AND   *
C*    PERROR(9) ARE DEFINED SOLELY FOR PROGRAMMING EASE. THEY ARE NOT  *
C*    USED.                                                            *
C***********************************************************************
C
      DATA PERROR/  1.0, 1.0, 1.92857, 2.78161, 3.56735,
     +              4.29497, 4.9065, 5.6066, 1.0/
C
C***********************************************************************
C*    THE COEFFICIENTS IN THE ARRAY PC ARE USED BOTH IN THE CONVERGENCE *
C*    AND ERROR TESTS. THEY ARE THE RECIPROCAL VALUES OF THE           *
C*    DISCRETIZATION ERROR CONSTANTS FOR EQUATIONS CONSTITUTING THE    *
C*    METHODS OF ORDER 1 THRU 7.                                       *
C***********************************************************************
C
      DATA PC/  2.0, 4.5, 7.3333, 6.0, 10.4167, 9.3, 13.7, 13.8687,
     +          9.6904, 17.15, 16.9504, 17.4349, 9.472, 20.7429, 15.921,
     +          14.7809/
C
C***********************************************************************
C*    THE COEFFICIENTS APPEARING IN THE ARRAY PD ARE USED IN THE TESTING*
C*    OF THE *OUTDATEDNESS* OF THE ARRAY RW.  THE PD(I) ELEMENT CONTAINS*
C*    THE AVERAGE VALUE OF THE COEFFICIENTS IN ARRAY C CORRESPONDING TO *
C*    ORDER I.                                                         *
C***********************************************************************
C
      DATA PD/  1.0, 0.6667, 0.6061, 0.4981, 0.4644, 0.4368, 0.412/
C
C***********************************************************************
C***********************************************************************
C
      KFLAG = 0
      IFAIL = 0
      IF(JSTART.NE.0) GO TO 80
C*                                                                     *
C*         INITIALIZATION --- FIRST CALL                               *
C*                                                                     *
   20 N0 = N
      FN=DBLE(FLOAT(N))
      N9 = N0 * 9
      N9P1 = N9 + 1
      N10 = N9 + N0
      N11 = N10 + N0
      N11P1 = N11 + 1
      N12 = N11 + N0
      N12P1 = N12 + 1
      NSQ = N0*N0
      IWEVAL=+1
      TDL=T
      H=DMAX1(DABS(HMIN),DMIN1(DABS(HNEXT),DABS(HMAX)))
      IF(HNEXT.LT.0.D0) H=-H
   30 NQOLD= 0
      ISW1=0
      ISW2=0
      CRATE=1.D0
      CALL DIFFUN (N, T, Y, DY)
      NFE=NFE+1
      DO 40 I=1,N
   40   DY(I,1) = H*DY(I,1)
      NQ = 1
      IST = 1
      IDEL = 0
      GO TO 180
```

```
C*                                                                      *STNT385Ø
C*         CONTINUE WITH THE STEP-SIZE H                                *STNT386Ø
C*                                                                      *STNT387Ø
   8Ø HNEW=DMAX1(DABS(HMIN),DMIN1(DABS(HNEW),DABS(HNEXT),DABS(HMAX)))    STNT388Ø
      IF(H.LT.Ø.DØ) HNEW=-HNEW                                          STNT389Ø
      IF(H.NE.HNEW) GO TO 1ØØ                                           STNT39ØØ
      IST = NQP1                                                        STNT391Ø
      IDEL = NQP1                                                       STNT392Ø
      ISW2=Ø                                                            STNT393Ø
      GO TO 18Ø                                                         STNT394Ø
C*                                                                      *STNT395Ø
C*         NEW STEP-SIZE --- INTERPOLATE FOR NEW POINTS                 *STNT396Ø
C*                                                                      *STNT397Ø
  1ØØ RATIO=HNEW/H                                                      STNT398Ø
      H = RATIO*H                                                       STNT399Ø
      RC=RC*RATIO*DBLE(PD(NQ)/PDOLD)                                    STNT4ØØØ
      PDOLD = PD(NQ)                                                    STNT4Ø1Ø
      IF (NQ.EQ.1) GO TO 15Ø                                            STNT4Ø2Ø
      IEQ = NEQ                                                         STNT4Ø3Ø
      D = Ø.ØDØ                                                         STNT4Ø4Ø
      DO 14Ø J=2,NQ                                                     STNT4Ø5Ø
        D = D + RATIO                                                   STNT4Ø6Ø
        IF (D.GT.DBLE(FLOAT(NEQ+NQP1-IEQ))) IEQ=2                       STNT4Ø7Ø
        D1 = DBLE(FLOAT(NEQ-IEQ-1)) - D                                 STNT4Ø8Ø
        DO 14Ø I=1,N                                                    STNT4Ø9Ø
          D2 = 1.ØDØ                                                    STNT41ØØ
          D3 = Ø.ØDØ                                                    STNT411Ø
          DO 12Ø J1=2,NQP1                                              STNT412Ø
            D2 = D2*(DBLE(FLOAT(J1)) + D1)/DBLE(FLOAT(J1-1))            STNT413Ø
  12Ø       D3 = D3 + D2*Y(I,J1,IEQ)                                    STNT414Ø
  14Ø     Y(I,J,1) = D3 + Y(I,1,IEQ)                                    STNT415Ø
  15Ø IST = NQ                                                          STNT416Ø
      IDEL = Ø                                                          STNT417Ø
      DO 16Ø I=1,N                                                      STNT418Ø
  16Ø   DY(I,1) = DY(I,1)*RATIO                                         STNT419Ø
      IRET = 3                                                          STNT42ØØ
      GO TO 4ØØØ                                                        STNT421Ø
C*                                                                      *STNT422Ø
C*         INITIALIZE SAVE ARRAY                                        *STNT423Ø
C*                                                                      *STNT424Ø
  18Ø DO 2ØØ I=1,N                                                      STNT425Ø
        SAVE(9,I) = DY(I,1)                                             STNT426Ø
        DO 2ØØ J=1,IST                                                  STNT427Ø
  2ØØ     SAVE(J,I) = Y(I,J,1)                                          STNT428Ø
      NQST = NQ                                                         STNT429Ø
      RATIO = 1.ØDØ                                                     STNT43ØØ
      TOLD = T                                                          STNT431Ø
      HOLD = H                                                          STNT432Ø
  22Ø IF ((NQ.EQ.NQOLD).AND.(FN.EQ.DBLE(FLOAT(N)))) GO TO 3ØØ           STNT433Ø
      IF ((NQ.EQ.NQOLD).AND.(FN.NE.DBLE(FLOAT(N)))) GO TO 28Ø           STNT434Ø
      IF (MAXORD.LT.8) GO TO 26Ø                                        STNT435Ø
      KFLAG = -7                                                        STNT436Ø
      GO TO 4185                                                        STNT437Ø
C*                                                                      *STNT438Ø
C*         SET APPROPRIATE PARAMETERS AND CONSTANTS                     *STNT439Ø
C*         FOR NEW CYCLE OF ORDER NQ                                    *STNT44ØØ
C*                                                                      *STNT441Ø
  26Ø INDB = INDEX(NQ,1)                                                STNT442Ø
      INDC = INDEX(NQ,2)                                                STNT443Ø
      NEQ = 3+NQ/5                                                      STNT444Ø
      JSTART = NQ                                                       STNT445Ø
      NQOLD = NQ                                                        STNT446Ø
      NQM1=NQ-1                                                         STNT447Ø
      NQP1 = NQ+1                                                       STNT448Ø
      Q = DBLE(FLOAT(NQ))                                               STNT449Ø
      ENQDWN = Ø.5DØ/Q                                                  STNT45ØØ
      ENQSAM = Ø.5DØ/(Q+1.ØDØ)                                          STNT451Ø
      ENQUP = Ø.5DØ/(Q+2.ØDØ)                                           STNT452Ø
  28Ø FN = DBLE(FLOAT(N))                                               STNT453Ø
      EDOWN = FN*(DBLE(PERROR(NQ))*EPS)**2                              STNT454Ø
      EUP = FN*(DBLE(PERROR(NQ+2))*EPS)**2                              STNT455Ø
      ES = FN*(DBLE(PERROR(NQP1))*EPS)**2                               STNT456Ø
      IF(EDOWN.GT.Ø.DØ) GO TO 3ØØ                                       STNT457Ø
      KFLAG = -5                                                        STNT458Ø
      GO TO 4185                                                        STNT459Ø
C*                                                                      *STNT46ØØ
```

```
C*        CHECK FOR REEVALUATION OF JACOBIAN                          *STNT461Ø
C*                                                                    *STNT462Ø
  3ØØ IF(IWEVAL.GT.Ø) GO TO 32Ø                                       STNT463Ø
      IF(DABS(RC-1.DØ).GE.4.D-1) IWEVAL = +2                          STNT464Ø
      IF((TDL-TOLD)*H.GT.Ø.DØ) GO TO 32Ø                              STNT465Ø
      IF(DABS(RC-1.DØ).GE.8.D-1) IWEVAL = +1                          STNT466Ø
  32Ø INDBB = INDB                                                    STNT467Ø
      INDCC = INDC                                                    STNT468Ø
      DO 1ØØØ IEQ=1,NEQ                                               STNT469Ø
        IST = MOD (IEQ, NEQ) + 1                                      STNT47ØØ
        T = T + H                                                     STNT471Ø
        BND=FN*(DBLE(PC(INDCC))*ENQUP*EPS)**2                         STNT472Ø
        E=ES                                                          STNT473Ø
        IF(IEQ.GT.2) E=FN*(DBLE(PC(INDCC))*EPS)**2                    STNT474Ø
C*                                                                    *STNT475Ø
C*        PREDICT Y AND DY FOR THE NEXT MESH POINT                    *STNT476Ø
C*                                                                    *STNT477Ø
        DO 46Ø I=1,N                                                  STNT478Ø
          D = DY(I,IEQ)                                               STNT479Ø
          D1 = Y(I,1,IEQ) + Q*D                                       STNT48ØØ
          D2 = B(INDBB+NQM1)*D                                        STNT481Ø
          IF (NQ-2) 44Ø, 4ØØ, 34Ø                                     STNT482Ø
  34Ø     IF (IEQ-3) 4ØØ, 38Ø, 36Ø                                    STNT483Ø
  36Ø     D2 = D2 + B(INDBB+NQP1)*DY(I,3)                             STNT484Ø
  38Ø     D2 = D2 + B(INDBB+NQ)*DY(I,2)                               STNT485Ø
  4ØØ     DO 42Ø J=2,NQ                                               STNT486Ø
            D = Y(I,J,IEQ)                                            STNT487Ø
            D3 = DBLE(FLOAT(J-1))                                     STNT488Ø
            D1 = D1 + D*(D3-Q)/D3                                     STNT489Ø
  42Ø       D2 = D2 + D*B(INDBB+J-2)                                  STNT49ØØ
  44Ø     Y(I,1,IST) = D1                                             STNT491Ø
  46Ø     DY(I,IST) = D2                                              STNT492Ø
        IF(NQ.LE.2) GO TO 47Ø                                         STNT493Ø
        IF(IEQ.GT.1) INDBB = INDBB + NQ + IEQ - 2                     STNT494Ø
C*                                                                    *STNT495Ø
C*        ITERATE THE CORRECTOR UP TO THREE TIMES. ACCUMULATE THE     *STNT496Ø
C*        CORRECTION TERMS IN SAVE(N1Ø+I,1) FOR REDOING THE ENTIRE    *STNT497Ø
C*        CORRECTOR IF CONVERGENCE IS NOT ACHIEVED                    *STNT498Ø
C*                                                                    *STNT499Ø
  47Ø   D1 = C(INDCC)                                                 STNT5ØØØ
  5ØØ   DO 51Ø I=1,N                                                  STNT5Ø1Ø
  51Ø     SAVE(N1Ø+I,1) = Ø.DØ                                        STNT5Ø2Ø
        DO 78Ø J=1,3                                                  STNT5Ø3Ø
          CALL DIFFUN (N, T, Y(1,1,IST), SAVE(N9P1,1))                STNT5Ø4Ø
          NFE=NFE+1                                                   STNT5Ø5Ø
          IF(IWEVAL-1) 68Ø, 52Ø, 62Ø                                  STNT5Ø6Ø
  52Ø     IND = 1                                                     STNT5Ø7Ø
          IF(IEQ.EQ.2) IND = 1                                        STNT5Ø8Ø
          GO TO (61Ø, 53Ø), MF                                        STNT5Ø9Ø
  53Ø     CALL DIFFUN (N, T-H*DBLE(FLOAT(1+IEQ-IND)), Y(1,1,IND),     STNT51ØØ
     +      SAVE(N12P1,1))                                           STNT511Ø
          NFE=NFE+1                                                   STNT512Ø
          D = Ø.DØ                                                    STNT513Ø
          DO 54Ø I=1,N                                                STNT514Ø
  54Ø       D = D + SAVE(N12+I,1)**2                                  STNT515Ø
          D = DABS(H)*1.D3*UROUND*DSQRT(D)                            STNT516Ø
          NJE = NJE + 1                                               STNT517Ø
          JØ = Ø                                                      STNT518Ø
          DO 56Ø J1=1,N                                               STNT519Ø
            DI = SQRTUR*YMAX(J1)                                      STNT52ØØ
            DI = DMAX1(DI,D)                                          STNT521Ø
            YJ1 = Y(J1,1,IND)                                         STNT522Ø
            Y(J1,1,IND) = Y(J1,1,IND) + DI                            STNT523Ø
            CALL DIFFUN (N, T-H*DBLE(FLOAT(1+IEQ-IND)), Y(1,1,IND),   STNT524Ø
     +        SAVE(N11P1,1))                                          STNT525Ø
            NFE = NFE + 1                                             STNT526Ø
            DO 55Ø I=1,N                                              STNT527Ø
  55Ø         RJ(I+JØ) = (SAVE(N11+I,1) - SAVE(N12+I,1))/DI           STNT528Ø
            JØ = JØ + NØ                                              STNT529Ø
            Y(J1,1,IND) = YJ1                                         STNT53ØØ
  56Ø       CONTINUE                                                  STNT531Ø
          GO TO 62Ø                                                   STNT532Ø
  61Ø     CALL PEDERV(N, T-H*DBLE(FLOAT(1+IEQ-IND)), Y(1,1,IND), RJ, NØ)STNT533Ø
          NJE=NJE+1                                                   STNT534Ø
  62Ø     D = D1*H                                                    STNT535Ø
          DO 64Ø I=1,NSQ                                              STNT536Ø
```

```
 64Ø       RW(I) = RJ(I)*D                                                  STNT537Ø
          J1 = 1                                                           STNT538Ø
          DO 66Ø I=1,N                                                     STNT539Ø
            RW(J1) = 1.DØ + RW(J1)                                         STNT54ØØ
 66Ø        J1 = J1 + NØ + 1                                               STNT541Ø
          CALL DEC (N, NØ, RW, IPIV, IER)                                  STNT542Ø
          NINVS=NINVS+1                                                    STNT543Ø
          IWEVAL = -IEQ                                                    STNT544Ø
          RC=1.DØ                                                          STNT545Ø
          PDOLD=PD(NQ)                                                     STNT546Ø
          IF(IER.NE.Ø) GO TO 8ØØ                                           STNT547Ø
 68Ø      DO 7ØØ I=1,N                                                     STNT548Ø
 7ØØ        SAVE(N9+I,1) = DY(I,IST) - H*SAVE(N9+I,1)                      STNT549Ø
          CALL SOL (N, NØ, RW, SAVE(N9P1,1), IPIV)                         STNT55ØØ
          D2 = Ø.ØDØ                                                       STNT551Ø
          J3 = N9                                                          STNT552Ø
          DO 76Ø I=1,N                                                     STNT553Ø
            J3 = J3 + 1                                                    STNT554Ø
            J2 = J3 + NØ                                                   STNT555Ø
            SAVE(J2,1) = SAVE(J2,1) + SAVE(J3,1)                           STNT556Ø
            Y(I,1,IST) = Y(I,1,IST) + D1*SAVE(J3,1)                        STNT557Ø
            DY(I,IST) = DY(I,IST) - SAVE(J3,1)                             STNT558Ø
 76Ø        D2 = D2 + (SAVE(J3,1)/YMAX(I))**2                              STNT559Ø
          IF(J.NE.1) CRATE = DMAX1(CRATE*9.D-1,D2/D3)                      STNT56ØØ
          IF (D2*DMIN1(1.DØ,2.DØ*CRATE).GT.BND) GO TO 77Ø                  STNT561Ø
          GO TO 94Ø                                                        STNT562Ø
 77Ø      D3 = D2                                                          STNT563Ø
 78Ø      CONTINUE                                                         STNT564Ø
C*                                                                        *STNT565Ø
C*          CORRECTOR FAILED TO CONVERGE IN THREE ITERATIONS.             *STNT566Ø
C*          IF JACOBIAN WAS REEVALUATED DURING THIS CYCLE, STEP-SIZE IS   *STNT567Ø
C*          REDUCED TO 3/1Ø OF H. OTHERWISE, JACOBIAN IS REEVALUATED      *STNT568Ø
C*                                                                        *STNT569Ø
        TDL = TOLD                                                         STNT57ØØ
        IF(IWEVAL.EQ.Ø) GO TO 9ØØ                                          STNT571Ø
 8ØØ    IF(DABS(H).LE.(1.ØØØØ1DØ*DABS(HMIN))) IF(NQ-2) 418Ø, 1Ø8Ø, 414Ø  STNT572Ø
        RATIO = RATIO*Ø.3DØ                                                STNT573Ø
        IRET = 1                                                           STNT574Ø
        ISW1=Ø                                                             STNT575Ø
        ISW2=1                                                             STNT576Ø
        IWEVAL = +2                                                        STNT577Ø
        GO TO 3ØØØ                                                         STNT578Ø
 9ØØ    DO 92Ø I=1,N                                                      STNT579Ø
          D = SAVE(N1Ø+I,1)                                                STNT58ØØ
          Y(I,1,IST) = Y(I,1,IST) - D1*D                                   STNT581Ø
 92Ø      DY(I,IST) = DY(I,IST) + D                                        STNT582Ø
        IWEVAL = +1                                                        STNT583Ø
        GO TO 5ØØ                                                          STNT584Ø
C*                                                                        *STNT585Ø
C*          CORRECTOR CONVERGED. THE BACKWARD DIFFERENCES OF ORDER        *STNT586Ø
C*          ONE THROUGH NQ AT THE NEW MESH POINT ARE COMPUTED             *STNT587Ø
C*                                                                        *STNT588Ø
 94Ø    DO 98Ø I=1,N                                                       STNT589Ø
          DO 96Ø J=1,NQ                                                    STNT59ØØ
 96Ø        Y(I,J+1,IST) = Y(I,J,IST) - Y(I,J,IEQ)                         STNT591Ø
        IF(IEQ.EQ.1) GO TO 98Ø                                             STNT592Ø
C*                                                                        *STNT593Ø
C*          THE (NQ+1)-ST BACKWARD DIFFERENCE FOR ALL BUT THE FIRST       *STNT594Ø
C*          MESH POINT IS ESTABLISHED                                     *STNT595Ø
C*                                                                        *STNT596Ø
        SAVE(N9+I,1) = Y(I,NQP1,IST) - Y(I,NQP1,IEQ)                       STNT597Ø
 98Ø    CONTINUE                                                           STNT598Ø
        IF (IEQ.EQ.NEQ) GO TO 1Ø15                                         STNT599Ø
        IF (NQ.LE.2) GO TO 1Ø1Ø                                            STNT6ØØØ
        IF  ((IEQ.GT.1).OR.(NQ.EQ.6)) INDCC = INDCC + 1                    STNT6Ø1Ø
 1Ø1Ø   IF(IEQ.EQ.1) GO TO 1ØØØ                                            STNT6Ø2Ø
C*                                                                        *STNT6Ø3Ø
C*          ERROR TEST FOR ALL BUT THE FIRST MESH POINT IS PERFORMED      *STNT6Ø4Ø
C*                                                                        *STNT6Ø5Ø
 1Ø15   D = Ø.DØ                                                           STNT6Ø6Ø
        DO 1Ø2Ø I=1,N                                                      STNT6Ø7Ø
 1Ø2Ø     D = D + (SAVE(N9+I,1)/YMAX(I))**2                               STNT6Ø8Ø
        IF(D.GT.E) GO TO 1Ø3Ø                                              STNT6Ø9Ø
 1ØØØ   CONTINUE                                                           STNT61ØØ
      GO TO 116Ø                                                           STNT611Ø
C*                                                                        *STNT612Ø
```

```
C*        THE ERROR CRITERION WAS NOT MET                                    *STNT613Ø
C*                                                                           *STNT614Ø
 1Ø3Ø IFAIL=IFAIL+1                                                           STNT615Ø
      IF(IFAIL.GT.2) GO TO 1Ø8Ø                                             STNT616Ø
      IF(DABS(H).LE.(DABS(HMIN)*1.ØØØØ1DØ)) IF(NQ-2) 114Ø, 412Ø, 414Ø       STNT617Ø
      IWEVAL = +2                                                           STNT618Ø
      IF(IFAIL.EQ.1) GO TO 12ØØ                                             STNT619Ø
      TDL=T                                                                 STNT62ØØ
      RATIO = RATIO*5.D-1                                                   STNT621Ø
      IRET = 1                                                              STNT622Ø
      ISW1=Ø                                                                STNT623Ø
      ISW2=1                                                                STNT624Ø
      GO TO 3ØØØ                                                            STNT625Ø
C*                                                                          *STNT626Ø
C*        START OVER WITH ORDER 1 METHOD                                    *STNT627Ø
C*                                                                          *STNT628Ø
 1Ø8Ø IFAIL = Ø                                                             STNT629Ø
      DO 1Ø9Ø I=1,N                                                         STNT63ØØ
 1Ø9Ø   Y(I,1,1) = SAVE(1,I)                                                STNT631Ø
      T = TOLD                                                              STNT632Ø
      H = H*DMAX1(1.D-1,DABS(HMIN/H))                                       STNT633Ø
      IWEVAL = +2                                                           STNT634Ø
      GO TO 3Ø                                                              STNT635Ø
C*                                                                          *STNT636Ø
C*        A NEW BLOCK OF SOLUTION POINTS HAS BEEN ACCEPTED                  *STNT637Ø
C*                                                                          *STNT638Ø
 114Ø IWEVAL = Ø                                                            STNT639Ø
      NSTEP = NSTEP + IEQ                                                   STNT64ØØ
      KFLAG = -IEQ                                                          STNT641Ø
      RETURN                                                                STNT642Ø
 116Ø IWEVAL = Ø                                                            STNT643Ø
      E = ES                                                                STNT644Ø
      KFLAG = NEQ                                                           STNT645Ø
      NSTEP = NSTEP + KFLAG                                                 STNT646Ø
      HNEW=H                                                                STNT647Ø
C*                                                                          *STNT648Ø
C*        CHECK FOR CONTINUATION WITH THE SAME H AND NQ                     *STNT649Ø
C*                                                                          *STNT65ØØ
      IF(ISW2.EQ.1) GO TO 142Ø                                              STNT651Ø
      IF(NQ.GT.3) ISW1=1-ISW1                                               STNT652Ø
      IF(ISW1.EQ.1) GO TO 142Ø                                              STNT653Ø
C*                                                                          *STNT654Ø
C*        NEW STEP-SIZE AND/OR ORDER SELECTION                              *STNT655Ø
C*                                                                          *STNT656Ø
 12ØØ RRSAME=1.2DØ*(D/E)**ENQSAM                                            STNT657Ø
      IF(IFAIL.NE.Ø) GO TO 138Ø                                             STNT658Ø
      RMAX = 1.D-4                                                          STNT659Ø
      DF=DBLE(FLOAT(NEQ+NQM1))                                              STNT66ØØ
      IF(NQ.NE.1) RMAX=(Q-1.DØ)/DF                                          STNT661Ø
      RRSAME=DMAX1(RRSAME,RMAX)                                             STNT662Ø
      RRUP = 1.D2Ø                                                          STNT663Ø
      RRDOWN = 1.D2Ø                                                        STNT664Ø
      IF(NQ.GE.MAXORD) GO TO 124Ø                                           STNT665Ø
      D = Ø.ØDØ                                                             STNT666Ø
      DO 122Ø I=1,N                                                         STNT667Ø
        D1 = Y(I,NQP1,NEQ) - Y(I,NQP1,NEQ-1)                                STNT668Ø
 122Ø   D = D + ((SAVE(N9+I,1) - D1)/YMAX(I))**2                            STNT669Ø
      RRUP = 1.2DØ*(D/EUP)**ENQUP                                           STNT67ØØ
      RMAX=Q/DF                                                             STNT671Ø
      RRUP=DMAX1(RRUP,RMAX)                                                 STNT672Ø
 124Ø IF(NQ.EQ.1) GO TO 128Ø                                                STNT673Ø
      D = Ø.ØDØ                                                             STNT674Ø
      DO 126Ø I=1,N                                                         STNT675Ø
 126Ø   D = D + (Y(I,NQP1,1)/YMAX(I))**2                                    STNT676Ø
      RRDOWN = 1.2DØ*(D/EDOWN)**ENQDWN                                      STNT677Ø
      RMAX = 1.D-4                                                          STNT678Ø
      IF(NQ.NE.2) RMAX=(Q-2.DØ)/DF                                          STNT679Ø
      RRDOWN=DMAX1(RRDOWN,RMAX)                                             STNT68ØØ
 128Ø IF (RRSAME.GT.RRUP) IF (RRUP-RRDOWN) 134Ø, 13ØØ, 13ØØ                 STNT681Ø
      IF (RRSAME.LE.RRDOWN) GO TO 132Ø                                      STNT682Ø
 13ØØ NEWQ = NQM1                                                           STNT683Ø
      D=1.DØ/RRDOWN                                                         STNT684Ø
      GO TO 136Ø                                                            STNT685Ø
 132Ø NEWQ = NQ                                                             STNT686Ø
      D=1.DØ/RRSAME                                                         STNT687Ø
      GO TO 136Ø                                                            STNT688Ø
```

```
 134Ø NEWQ = NQP1                                                        STNT689Ø
      D=1.DØ/RRUP                                                        STNT69ØØ
 136Ø IF (D-1.1DØ) 142Ø, 14ØØ, 14ØØ                                      STNT691Ø
 138Ø RATIO=RATIO/RRSAME                                                 STNT692Ø
      IRET = 1                                                           STNT693Ø
      ISW1=Ø                                                             STNT694Ø
      ISW2=1                                                             STNT695Ø
      GO TO 3ØØØ                                                         STNT696Ø
 14ØØ HNEW = H*D                                                         STNT697Ø
      NQ = NEWQ                                                          STNT698Ø
 142Ø DO 146Ø I=1,N                                                      STNT699Ø
        D=YMAX(I)                                                        STNT7ØØØ
        DO 144Ø J=1,NEQ                                                  STNT7Ø1Ø
 144Ø     D=DMAX1(D,DABS(Y(I,1,J)))                                      STNT7Ø2Ø
 146Ø   YMAX(I)=D                                                        STNT7Ø3Ø
      HNEXT = HNEW                                                       STNT7Ø4Ø
      KNEXT = 3+NQ/5                                                     STNT7Ø5Ø
      RETURN                                                             STNT7Ø6Ø
C*                                                                      *STNT7Ø7Ø
C*        THIS SECTION IS USED WHEN STEP-SIZE OR ORDER IS CHANGED       *STNT7Ø8Ø
C*        DURING THE CYCLE.  STARTING VALUES ARE RETRIEVED FROM THE     *STNT7Ø9Ø
C*        SAVE ARRAY.                                                   *STNT71ØØ
C*                                                                      *STNT711Ø
 3ØØØ RATIO = DMAX1 (DABS(HMIN/HOLD), DMIN1(RATIO, 1.ØDØ))               STNT712Ø
      T = TOLD                                                           STNT713Ø
      IF (RATIO.LT.1.ØDØ) IF (IDEL) 3Ø3Ø, 3Ø3Ø, 3Ø8Ø                     STNT714Ø
      DO 3Ø2Ø I=1,N                                                      STNT715Ø
        DY(I,1) = SAVE(9,I)                                              STNT716Ø
        DO 3Ø2Ø J=1,NQ                                                   STNT717Ø
 3Ø2Ø     Y(I,J,1) = SAVE(J,I)                                           STNT718Ø
      GO TO (32Ø,26Ø), IRET                                              STNT719Ø
C*                                                                      *STNT72ØØ
C*        THE (NQST+1)-ST ORDER BACKWARD DIFFERENCE IS ESTABLISHED      *STNT721Ø
C*                                                                      *STNT722Ø
 3Ø3Ø IDEL = NQST+1                                                      STNT723Ø
      IF ((NQST.LT.2) .OR. (NQ.LT.2)) GO TO 3Ø8Ø                         STNT724Ø
      D = DBLE(FLOAT(NQST))                                              STNT725Ø
      DO 3Ø6Ø I=1,N                                                      STNT726Ø
        D1 = SAVE(9,I)                                                   STNT727Ø
        DO 3Ø4Ø J=2,NQST                                                 STNT728Ø
 3Ø4Ø     D1 = D1 - SAVE(J,I)/DBLE(FLOAT(J-1))                           STNT729Ø
 3Ø6Ø   SAVE(IDEL,I) = D*D1                                              STNT73ØØ
C*                                                                      *STNT731Ø
C*        INTERPOLATE FOR NEW POINTS                                    *STNT732Ø
C*                                                                      *STNT733Ø
 3Ø8Ø DO 314Ø I=1,N                                                      STNT734Ø
        DY(I,1) = RATIO*SAVE(9,I)                                        STNT735Ø
        IF (NQ.LT.2) GO TO 314Ø                                          STNT736Ø
        D1 = Ø.ØDØ                                                       STNT737Ø
        DO 312Ø J=2,NQ                                                   STNT738Ø
          D1 = D1 + RATIO                                                STNT739Ø
          D2 = 1.ØDØ                                                     STNT74ØØ
          D = Ø.ØDØ                                                      STNT741Ø
          DO 31ØØ J1=2,IDEL                                              STNT742Ø
            D2 = D2*(DBLE(FLOAT(J1-2)) - D1)/DBLE(FLOAT(J1-1))           STNT743Ø
 31ØØ       D = D + D2*SAVE(J1,I)                                        STNT744Ø
 312Ø     Y(I,J,1) = D + SAVE(1,I)                                       STNT745Ø
 314Ø   Y(I,1,1) = SAVE(1,I)                                             STNT746Ø
      H = HOLD*RATIO                                                     STNT747Ø
C*                                                                      *STNT748Ø
C*        FORM THE BACKWARD DIFFERENCES                                 *STNT749Ø
C*                                                                      *STNT75ØØ
 4ØØØ IF (NQ.LT.2) GO TO 4Ø4Ø                                            STNT751Ø
      NQM1 = NQ-1                                                        STNT752Ø
      DO 4Ø2Ø I=1,N                                                      STNT753Ø
        DO 4Ø2Ø J=1,NQM1                                                 STNT754Ø
          JØ=J + 1                                                       STNT755Ø
          DO 4Ø2Ø J1=JØ,NQ                                               STNT756Ø
            J2 = NQ-J1+J+1                                               STNT757Ø
 4Ø2Ø       Y(I,J2,1) = Y(I,J2-1,1) - Y(I,J2,1)                          STNT758Ø
 4Ø4Ø GO TO (32Ø,26Ø,18Ø), IRET                                          STNT759Ø
C*                                                                      *STNT76ØØ
C*        CONVERGENCE OR ERROR PROBLEMS                                 *STNT761Ø
C*                                                                      *STNT762Ø
 412Ø NQ = 1                                                             STNT763Ø
      GO TO 416Ø                                                         STNT764Ø
```

```
 414Ø NQ = 2                                                              STNT765Ø
 416Ø IFAIL = Ø                                                           STNT766Ø
      IRET = 2                                                            STNT767Ø
      IWEVAL = +2                                                         STNT768Ø
      GO TO 3ØØØ                                                          STNT769Ø
 418Ø KFLAG = -6                                                          STNT77ØØ
 4185 J1 = NQST + 1                                                       STNT771Ø
      DO 419Ø I=1,N                                                       STNT772Ø
        DY(I,1) = SAVE(9,I)                                               STNT773Ø
        DO 419Ø J=1,J1                                                    STNT774Ø
 419Ø     Y(I,J,1) = SAVE(J,I)                                           STNT775Ø
      H = HOLD                                                            STNT776Ø
      T = TOLD                                                            STNT777Ø
      JSTART = NQST                                                       STNT778Ø
      RETURN                                                              STNT779Ø
C                                                                         STNT78ØØ
C*************************************************************************STNT781Ø
C*                                                                       *STNT782Ø
C*                    --- END OF SUBROUTINE STINT ---                    *STNT783Ø
C*                                                                       *STNT784Ø
C*************************************************************************STNT785Ø
C                                                                         STNT786Ø
      END                                                                 STNT787Ø


      SUBROUTINE DEC (N, NDIM, A, IP, IER)                                DECØ  1Ø
C                                                                         DECØ  2Ø
      INTEGER N,NDIM,IP,IER,NM1,K,KP1,M,I,J                               DECØ  3Ø
      DOUBLE PRECISION A, T                                               DECØ  4Ø
C     DOUBLE PRECISION DABS                                               DECØ  5Ø
      DIMENSION A(NDIM,N), IP(N)                                          DECØ  6Ø
C-------------------------------------------------------------------------DECØ  7Ø
C  MATRIX TRIANGULARIZATION BY GAUSSIAN ELIMINATION.                      DECØ  8Ø
C  INPUT..                                                                DECØ  9Ø
C     N = ORDER OF MATRIX.                                                DECØ 1ØØ
C     NDIM = DECLARED DIMENSION OF ARRAY  A .                             DECØ 11Ø
C     A = MATRIX TO BE TRIANGULARIZED.                                    DECØ 12Ø
C  OUTPUT..                                                               DECØ 13Ø
C     A(I,J), I.LE.J = UPPER TRIANGULAR FACTOR, U .                       DECØ 14Ø
C     A(I,J), I.GT.J = MULTIPLIERS = LOWER TRIANGULAR FACTOR, I - L.      DECØ 15Ø
C     IP(K), K.LT.N = INDEX OF K-TH PIVOT ROW.                            DECØ 16Ø
C     IP(N) = (-1)**(NUMBER OF INTERCHANGES) OR 0 .                       DECØ 17Ø
C     IER = Ø IF MATRIX A IS NONSINGULAR, OR K IF FOUND TO BE             DECØ 18Ø
C           SINGULAR AT STAGE K.                                          DECØ 19Ø
C  USE  SOL  TO OBTAIN SOLUTION OF LINEAR SYSTEM.                         DECØ 2ØØ
C  DETERM(A) = IP(N)*A(1,1)*A(2,2)*...*A(N,N).                            DECØ 21Ø
C  IF IP(N)=0, A IS SINGULAR, SOL WILL DIVIDE BY ZERO.                    DECØ 22Ø
C                                                                         DECØ 23Ø
C  REFERENCE..                                                            DECØ 24Ø
C     C. B. MOLER, ALGORITHM 423, LINEAR EQUATION SOLVER,                 DECØ 25Ø
C     C.A.C.M. 15 (1972), P. 274.                                         DECØ 26Ø
C-------------------------------------------------------------------------DECØ 27Ø
      IER = Ø                                                             DECØ 28Ø
      IP(N) = 1                                                           DECØ 29Ø
      IF (N .EQ. 1) GO TO 7Ø                                              DECØ 3ØØ
      NM1 = N - 1                                                         DECØ 31Ø
      DO 6Ø K = 1,NM1                                                     DECØ 32Ø
        KP1 = K + 1                                                       DECØ 33Ø
        M = K                                                             DECØ 34Ø
        DO 1Ø I = KP1,N                                                   DECØ 35Ø
 1Ø       IF (DABS(A(I,K)) .GT. DABS(A(M,K))) M = I                      DECØ 36Ø
        IP(K) = M                                                         DECØ 37Ø
        T = A(M,K)                                                        DECØ 38Ø
        IF (M .EQ. K) GO TO 2Ø                                            DECØ 39Ø
        IP(N) = -IP(N)                                                    DECØ 4ØØ
        A(M,K) = A(K,K)                                                   DECØ 41Ø
        A(K,K) = T                                                        DECØ 42Ø
 2Ø     IF (T .EQ. Ø.DØ) GO TO 8Ø                                         DECØ 43Ø
        T = 1.DØ/T                                                        DECØ 44Ø
        DO 3Ø I = KP1,N                                                   DECØ 45Ø
 3Ø       A(I,K) = -A(I,K)*T                                             DECØ 46Ø
        DO 5Ø J = KP1,N                                                   DECØ 47Ø
          T = A(M,J)                                                      DECØ 48Ø
          A(M,J) = A(K,J)                                                 DECØ 49Ø
          A(K,J) = T                                                      DECØ 5ØØ
          IF (T .EQ. Ø.DØ) GO TO 5Ø                                       DECØ 51Ø
```

```
          DO 4Ø I = KP1,N                                                    DECØ 52Ø
 4Ø         A(I,J) = A(I,J) + A(I,K)*T                                       DECØ 53Ø
 5Ø       CONTINUE                                                           DECØ 54Ø
 6Ø     CONTINUE                                                             DECØ 55Ø
 7Ø   K = N                                                                  DECØ 56Ø
      IF (A(N,N) .EQ. Ø.DØ) GO TO 8Ø                                         DECØ 57Ø
      RETURN                                                                 DECØ 58Ø
 8Ø   IER = K                                                                DECØ 59Ø
      IP(N) = Ø                                                              DECØ 6ØØ
      RETURN                                                                 DECØ 61Ø
C---------------------- END OF SUBROUTINE DEC ------------------------------DECØ 62Ø
      END                                                                    DECØ 63Ø

      SUBROUTINE SOL (N, NDIM, A, B, IP)                                     SOLØ  1Ø
C                                                                            SOLØ  2Ø
      INTEGER N,NDIM,IP,NM1,K,KP1,M,I,KB,KM1                                 SOLØ  3Ø
      DOUBLE PRECISION A,B,T                                                 SOLØ  4Ø
      DIMENSION A(NDIM,N), B(N), IP(N)                                       SOLØ  5Ø
C------------------------------------------------------------------------------SOLØ  6Ø
C  SOLUTION OF LINEAR SYSTEM, A*X = B .                                      SOLØ  7Ø
C  INPUT..                                                                   SOLØ  8Ø
C    N = ORDER OF MATRIX.                                                    SOLØ  9Ø
C    NDIM = DECLARED DIMENSION OF ARRAY  A .                                 SOLØ 1ØØ
C    A = TRIANGULARIZED MATRIX OBTAINED FROM DEC.                            SOLØ 11Ø
C    B = RIGHT HAND SIDE VECTOR.                                             SOLØ 12Ø
C    IP = PIVOT VECTOR OBTAINED FROM DEC.                                    SOLØ 13Ø
C  DO NOT USE IF DEC HAS SET IER .NE. Ø.                                     SOLØ 14Ø
C  OUTPUT..                                                                  SOLØ 15Ø
C    B = SOLUTION VECTOR, X .                                                SOLØ 16Ø
C------------------------------------------------------------------------------SOLØ 17Ø
      IF (N .EQ. 1) GO TO 5Ø                                                 SOLØ 18Ø
      NM1 = N - 1                                                            SOLØ 19Ø
      DO 2Ø K = 1,NM1                                                        SOLØ 2ØØ
        KP1 = K + 1                                                          SOLØ 21Ø
        M = IP(K)                                                            SOLØ 22Ø
        T = B(M)                                                             SOLØ 23Ø
        B(M) = B(K)                                                          SOLØ 24Ø
        B(K) = T                                                             SOLØ 25Ø
        DO 1Ø I = KP1,N                                                      SOLØ 26Ø
 1Ø       B(I) = B(I) + A(I,K)*T                                            SOLØ 27Ø
 2Ø     CONTINUE                                                            SOLØ 28Ø
      DO 4Ø KB = 1,NM1                                                       SOLØ 29Ø
        KM1 = N - KB                                                         SOLØ 3ØØ
        K = KM1 + 1                                                          SOLØ 31Ø
        B(K) = B(K)/A(K,K)                                                   SOLØ 32Ø
        T = -B(K)                                                            SOLØ 33Ø
        DO 3Ø I = 1,KM1                                                      SOLØ 34Ø
 3Ø       B(I) = B(I) + A(I,K)*T                                            SOLØ 35Ø
 4Ø     CONTINUE                                                            SOLØ 36Ø
 5Ø   B(1) = B(1)/A(1,1)                                                     SOLØ 37Ø
      RETURN                                                                 SOLØ 38Ø
C---------------------- END OF SUBROUTINE SOL ------------------------------SOLØ 39Ø
      END                                                                    SOLØ 4ØØ

      SUBROUTINE DIFFUN(N, T, Y, YDOT)                                       TEST  1Ø
      DOUBLE PRECISION T, T1, T2, W1, W2, W3, W4, S, Y, YDOT                 TEST  2Ø
      DIMENSION Y(N), YDOT(N)                                                TEST  3Ø
      T1=-Y(1)+Y(2)+Y(3)+Y(4)                                                TEST  4Ø
      T2=Y(1)-Y(2)+Y(3)+Y(4)                                                 TEST  5Ø
      W1=5.D-1*(T1*T1-T2*T2)                                                 TEST  6Ø
      W2=T1*T2                                                               TEST  7Ø
      W3=(Y(1)+Y(2)-Y(3)+Y(4))**2                                            TEST  8Ø
      W4=(Y(1)+Y(2)+Y(3)-Y(4))**2                                            TEST  9Ø
      S=Ø.DØ                                                                 TEST 1ØØ
      DO 1Ø I=1,3                                                            TEST 11Ø
   1Ø S=S-1.D-3*Y(I)                                                         TEST 12Ø
      S=S+1.D-3*Y(4)                                                         TEST 13Ø
      YDOT(1)=-2.5D-1*(9.8D2*Y(1)+1.Ø2D3*Y(2)-9.8D2*Y(3)+1.Ø2D3*Y(4)-S)      TEST 14Ø
     +   +1.25D-1*(-W1+W2+W3+W4)                                             TEST 15Ø
      YDOT(2)=-2.5D-1*(1.Ø2D3*Y(1)+9.8D2*Y(2)-1.Ø2D3*Y(3)+9.8D2*Y(4)-S)      TEST 16Ø
     +   +1.25D-1*( W1-W2+W3+W4)                                             TEST 17Ø
      YDOT(3)=-2.5D-1*(-1.Ø2D3*Y(1)-9.8D2*Y(2)+9.8D2*Y(3)-1.Ø2D3*Y(4)-S)     TEST 18Ø
     +   +1.25D-1*( W1+W2-W3+W4)                                             TEST 19Ø
      YDOT(4)=-2.5D-1*(9.8D2*Y(1)+1.Ø2D3*Y(2)-1.Ø2D3*Y(3)+9.8D2*Y(4)+S)      TEST 2ØØ
     +   +1.25D-1*( W1+W2+W3-W4)                                             TEST 21Ø
```

```
      RETURN                                                              TEST 22Ø
      END                                                                 TEST 23Ø
      SUBROUTINE PEDERV(N, T, Y, PØ, NØ)                                  TEST 24Ø
      DOUBLE PRECISION PØ, Y, T                                           TEST 25Ø
      DIMENSION PØ(16), Y(N)                                              TEST 26Ø
      PØ(1)=(-9.8ØØØ1D2+Y(1)+Y(3)+Y(4)+3.DØ*Y(2))*2.5D-1                  TEST 27Ø
      PØ(5)=(-1.Ø2ØØØ1D3+Y(2)-Y(3)-Y(4)+3.DØ*Y(1))*2.5D-1                 TEST 28Ø
      PØ(9)=(9.79999D2+Y(1)-Y(2)-Y(4)+3.DØ*Y(3))*2.5D-1                   TEST 29Ø
      PØ(13)=(-1.Ø19999D3+Y(1)-Y(2)-Y(3)+3.DØ*Y(4))*2.5D-1                TEST 3ØØ
      PØ(2)=PØ(5)                                                         TEST 31Ø
      PØ(6)=PØ(1)                                                         TEST 32Ø
      PØ(1Ø)=-PØ(13)                                                      TEST 33Ø
      PØ(14)=-PØ(9)                                                       TEST 34Ø
      PØ(3)=PØ(1Ø)                                                        TEST 35Ø
      PØ(7)=PØ(9)                                                         TEST 36Ø
      PØ(11)=PØ(1)                                                        TEST 37Ø
      PØ(15)=-PØ(5)                                                       TEST 38Ø
      PØ(4)=PØ(14)                                                        TEST 39Ø
      PØ(8)=PØ(13)                                                        TEST 4ØØ
      PØ(12)=PØ(15)                                                       TEST 41Ø
      PØ(16)=PØ(1)                                                        TEST 42Ø
      RETURN                                                              TEST 43Ø
      END                                                                 TEST 44Ø
      SUBROUTINE ERROR(T, Y, N, ERØ)                                      TEST 45Ø
      DOUBLE PRECISION T, T1, T2, T3, W1, W2, W3, W4, X, Y, ERØ           TEST 46Ø
C     DOUBLE PRECISION DEXP, DSIN, DCOS, DABS                             TEST 47Ø
      DIMENSION ERØ(N), Y(N)                                              TEST 48Ø
      X=1.D1*T                                                            TEST 49Ø
      IF(X.GT.1.65D2) GO TO 2Ø                                            TEST 5ØØ
      T1=1.DØ+DEXP(-X)*(9.DØ*DCOS(X)+1.D1*DSIN(X))                        TEST 51Ø
      T2=DEXP(-X)*(1.D1*DCOS(X)-9.ØDØ*DSIN(X))                            TEST 52Ø
      IF(DABS(T2).LE.1.D-15) GO TO 1Ø                                     TEST 53Ø
      T3=1.D1/(T1*T1+T2*T2)                                               TEST 54Ø
      GO TO 11                                                            TEST 55Ø
   1Ø T3=1.D1/T1**2                                                       TEST 56Ø
   11 W1=-(T1+T2)*T3                                                      TEST 57Ø
      W2=(T1-T2)*T3                                                       TEST 58Ø
      GO TO 21                                                            TEST 59Ø
   2Ø W1=-1.D1                                                            TEST 6ØØ
      W2=1.D1                                                             TEST 61Ø
   21 X=1.D3*T                                                            TEST 62Ø
      IF(X.GT.1.65D2) GO TO 22                                            TEST 63Ø
      W3=5.D2/(1.DØ-1.ØØ1D3*DEXP(X))                                      TEST 64Ø
      GO TO 23                                                            TEST 65Ø
   22 W3=Ø.DØ                                                             TEST 66Ø
   23 W4=5.D-4/(1.DØ-1.ØØ1DØ*DEXP(1.D-3*T))                               TEST 67Ø
      ERØ(1)=-W1+W2+W3+W4                                                 TEST 68Ø
      ERØ(2)= W1-W2+W3+W4                                                 TEST 69Ø
      ERØ(3)= W1+W2-W3+W4                                                 TEST 7ØØ
      ERØ(4)= W1+W2+W3-W4                                                 TEST 71Ø
      DO 3Ø I=1,N                                                         TEST 72Ø
   3Ø ERØ(I)=ERØ(I)-Y(I)                                                  TEST 73Ø
      RETURN                                                              TEST 74Ø
      END                                                                 TEST 75Ø
2Ø                                                                        TEST 76Ø
 6                                                                        TEST 77Ø
15                                                                        TEST 78Ø
Ø4                                                                        TEST 79Ø
 Ø.ØØØØØØØØØØØØØØØØD+ØØ-2.ØØØØØØØØØØØØØØØØD+ØØ-1.ØØØØØØØØØØØØØØØØD+ØØ  TEST 8ØØ
-1.ØØØØØØØØØØØØØØØØD+ØØ                                                  TEST 81Ø
 Ø.ØØØØØØØØØØØØØØØØD+ØØ 1.ØØØØØØØØØØØØØØØØD+Ø2 5.ØØØØØØØØØØØØØØØØD-Ø9  TEST 82Ø
 1                                                                        TEST 83Ø
 1.ØØØØØØØØØØØØØØØØD-Ø3                                                  TEST 84Ø
 1                                                                        TEST 85Ø
 1.ØØØØØØØØØØØØØØØØD-Ø5                                                  TEST 86Ø
 1                                                                        TEST 87Ø
 1.ØØØØØØØØØØØØØØØØD-Ø7                                                  TEST 88Ø
 2                                                                        TEST 89Ø
 1.ØØØØØØØØØØØØØØØØD-Ø3                                                  TEST 9ØØ
 2                                                                        TEST 91Ø
 1.ØØØØØØØØØØØØØØØØD-Ø5                                                  TEST 92Ø
 2                                                                        TEST 93Ø
 1.ØØØØØØØØØØØØØØØØD-Ø7                                                  TEST 94Ø
```

ALGORITHM 535
The QZ Algorithm To Solve the Generalized Eigenvalue Problem for Complex Matrices [F2]

BURTON S. GARBOW
Argonne National Laboratory

Key Words and Phrases: eigenvalues, generalized eigenvalue problem
CR Categories: 5.14
Language: Fortran

DESCRIPTION

Three Fortran subroutines are provided that implement a complex form of the QZ algorithm for finding λ and z such that $Az = \lambda Bz$, where A and B are complex N by N matrices. The subroutines are complex analogs of those for the corresponding real problem included in the EISPACK eigensystem package [1]; the original QZ algorithm is described in [3] and [4]. The complex QZ algorithm shares the fundamental property of the real QZ algorithm in being unaffected by singularity or near singularity of B.

Subroutine CQZHES implements the first step of the algorithm wherein A and B are simultaneously reduced by unitary transformations to upper Hessenberg and upper triangular form, respectively. Subroutine CQZVAL implements an iterative process that reduces A to upper triangular form while maintaining the triangular form of B. At the completion of this step, the eigenvalues are derivable from the corresponding diagonal elements of the reduced A and B. Finally, if eigenvectors are desired, subroutine CQZVEC applies the accumulated transformations from the two earlier steps onto the eigenvectors of the triangular problem.

No facility is provided for obtaining just a few eigenvectors; a subroutine employing inverse iteration might be included for this purpose, proceeding from the Hessenberg A, the triangular B, and an approximate eigenvalue of the problem. Inverse iteration could be expected to save time, but it might be somewhat less accurate and reliable, especially in the presence of close eigenvalues.

Also, no facility is included for balancing A and B. Optimally scaled matrices could result in further improvement of accuracy obtainable with unitary transformations, but no automatic scaling technique is known at this time.

The subroutines can be invoked with consecutive statements:

```
CALL CQZHES (NM, N, AR, AI, BR, BI, MATZ, ZR, ZI)
CALL CQZVAL (NM, N, AR, AI, BR, BI, EPS1, ALFR, ALFI, BETA, MATZ, ZR, ZI,
  IERR)
```

Received 18 May 1977; revised 10 November 1977.
This work was performed under the auspices of the U.S. Department of Energy.
Author's address: Applied Mathematics Division, Argonne National Laboratory, 9700 South Cass Ave., Argonne, IL 60439.

```
THE FULL MATRIX  A   OF ORDER   5  IS (PRINTED BY ROWS)

-238. -344.I      86.  178.I     164.  240.I    -166. -308.I      56.  158.I
  76.  152.I     -96. -128.I      40.  -32.I      60.  184.I     -60. -136.I
 118.  284.I      55. -182.I     -13.  460.I      34. -192.I     -176. -214.I
-314. -160.I     132.   78.I     114.  296.I     -90. -164.I     -424. -374.I
 -54.  -24.I    -205. -400.I     109.  148.I     158.  312.I     -38.  -96.I

THE FULL MATRIX  B   OF ORDER   5  IS (PRINTED BY ROWS)

 388.   94.I    -386. -122.I    -250.  -14.I     556.  130.I    -396.  -62.I
-304.  -76.I     384.   64.I    -160.   16.I    -240.  -92.I     240.   68.I
-658. -136.I     -73.  100.I    -109. -250.I    -118.  100.I     406.   96.I
-640.  -10.I     204.  -42.I    -692.  -90.I     288.   66.I    -192.  154.I
-162.  -72.I     631.  158.I     131.   52.I    -758. -184.I     278.   76.I

                ALFR(I)                    ALFI(I)                    BETA(I)
1           3.299829E 02              4.061311E 02              4.315166E 02
2          -1.368646E 02             -1.824827E 02              1.368646E 02
3          -5.341594E 02             -6.231853E 02              1.513450E 03
4          -1.905912E 02             -2.223549E 02              5.400093E 02
5          -9.995206E 01              1.166093E 02              2.831985E 02

              COMPUTED EIGENVALUE AND EIGENVECTOR                    RESIDUAL

1           7.647050E-01              9.411714E-01                  7.46E-07
             -5.370591E-01             -2.236047E-02
             -3.185631E-01             -1.325891E-02
             -3.713154E-01             -1.545933E-02
             -5.810177E-01             -2.418823E-02
             -9.991344E-01             -4.160066E-02

2          -1.000000E 00             -1.333308E 00                  4.84E-07
             -2.592583E-01             -8.606803E-02
              5.948840E-01              1.974900E-01
              8.487330E-01              2.817618E-01
              9.490676E-01              3.150727E-01
              2.346767E-01              7.791054E-02

3          -3.529415E-01             -4.117646E-01                  6.39E-07
              2.828103E-02              2.718275E-01
             -4.315349E-01             -4.140894E-01
              1.024388E-02              3.613692E-01
              3.964370E-01              9.180620E-01
             -2.121114E-01             -5.137083E-01

4          -3.529405E-01             -4.117613E-01                  4.82E-07
              5.404448E-01              5.624916E-02
              9.812239E-01              1.928717E-01
              8.455181E-01              9.440637E-02
              4.273624E-01             -2.578912E-02
             -2.842883E-01              7.476088E-03

5          -3.529399E-01              4.117583E-01                  7.78E-07
              1.844878E-03              9.193128E-02
              1.966345E-02              9.801325E-01
              1.144641E-02              5.705664E-01
              2.005622E-02              9.997990E-01
              3.575820E-03              1.782405E-01
```

Fig. 1

and, if IERR is zero and eigenvectors are desired:

CALL CQZVEC (NM, N, AR, AI, BR, BI, ALFR, ALFI, BETA, ZR, ZI)

where parameter usage is explained in the algorithm that follows.

In timing the subroutines on the IBM 370/195, it was found that they run between two and three times slower than the corresponding real subroutines for systems of the same order. This is felt to be a quite favorable result and is attributable chiefly to two factors: (1) double shifts used with real systems to avoid complex arithmetic can be replaced by more efficient single shifts here, and (2) the amount of complex arithmetic itself can be reduced by choosing one of the defining elements of each 2×2 unitary transformation to be real.

The subroutines provide an alternative to the LZ algorithm developed by Kaufman [2] for the solution of the complex generalized eigenproblem. The elementary transformations employed in the LZ algorithm, even though stabilized, do not preclude possible numerical growth with consequent loss of accuracy; the unitary transformations employed in the QZ algorithm, on the other hand,

are stable. Although unitary transformations require more arithmetic than elementary transformations, speed comparisons between the algorithms are complicated by the fact that the LZ algorithm employs complex operands while the QZ algorithm employs programmed complex arithmetic on real operands. With certain CDC compilers, for example, it has been demonstrated that formal complex arithmetic is more efficient than programmed complex arithmetic, thereby strengthening the advantage of the LZ algorithm; in contrast, with the H compiler on the IBM 370/195, the reverse is true: the complex QZ algorithm actually runs faster than does the LZ algorithm.

A long precision IBM version of the subroutines was first written and tested on the 370/195 at Argonne National Laboratory; the standard single precision Fortran version was then derived from it and also tested on the 370/195. There are no machine-dependent constants in the subroutines, so the standard version should run directly on different machines.

The example chosen for illustration is the same as that used in [2] and appears here as Figure 1. The quantities labeled RESIDUAL are computed as

$$\| \beta_i A z_i - \alpha_i B z_i \|_1 / (\| z_i \|_1 * [\beta_i \| A \|_1 + \| \alpha_i \| \| B \|_1])$$

where β_i = BETA(I), α_i = CMPLX(ALFR(I), ALFI(I)), and z_i = CMPLX(ZR (*, I), ZI (*, I)).

REFERENCES

1. GARBOW, B.S., BOYLE, J.M., DONGARRA, J.J., AND MOLER, C.B. Matrix eigensystem routines—EISPACK guide extension. *Lecture Notes in Computer Science, Vol. 51,* Springer-Verlag, Heidelberg, 1977.
2. KAUFMAN, L.C. Algorithm 496. The LZ algorithm to solve the generalized eigenvalue problem for complex matrices. *ACM Trans. Math. Software 1,* 3 (Sept. 1975), 271-281.
3. MOLER, C.B., AND STEWART, G.W. An algorithm for generalized matrix eigenvalue problems. *SIAM J. Numer. Anal. 10* (April 1973), 241-256.
4. WARD, R.C. The combination shift QZ algorithm. *SIAM J. Numer. Anal. 12* (Dec. 1975), 835-853.

ALGORITHM

NAME(n): indicates a Fortran module with n records
NAME^T: indicates "NAME" is included for testing purposes
NAME^D: indicates "NAME" contains test data
Contents: CQZHES(281), CQZVAL(382), CQZVEC(138), TEST^T(75), RMATIN^T(65), CGGWZR^T(120), DATA1^D(37), DATA2^D(36)

```
C                                                                       HES   10
C     ------------------------------------------------------------------HES   20
C                                                                       HES   30
      SUBROUTINE CQZHES(NM,N,AR,AI,BR,BI,MATZ,ZR,ZI)                    HES   40
C                                                                       HES   50
      INTEGER I,J,K,L,N,K1,LB,L1,NM,NK1,NM1                             HES   60
      REAL AR(NM,N),AI(NM,N),BR(NM,N),BI(NM,N),ZR(NM,N),ZI(NM,N)        HES   70
      REAL R,S,T,TI,U1,U2,XI,XR,YI,YR,RHO,U1I                           HES   80
      REAL SQRT,CABS,ABS                                                HES   90
      LOGICAL MATZ                                                      HES  100
      COMPLEX CMPLX                                                     HES  110
C                                                                       HES  120
C     THIS SUBROUTINE IS A COMPLEX ANALOGUE OF THE FIRST STEP OF THE    HES  130
C     QZ ALGORITHM FOR SOLVING GENERALIZED MATRIX EIGENVALUE PROBLEMS,  HES  140
C     SIAM J. NUMER. ANAL. 10, 241-256(1973) BY MOLER AND STEWART.      HES  150
C                                                                       HES  160
C     THIS SUBROUTINE ACCEPTS A PAIR OF COMPLEX GENERAL MATRICES AND    HES  170
C     REDUCES ONE OF THEM TO UPPER HESSENBERG FORM WITH REAL (AND NON-  HES  180
C     NEGATIVE) SUBDIAGONAL ELEMENTS AND THE OTHER TO UPPER TRIANGULAR  HES  190
C     FORM USING UNITARY TRANSFORMATIONS.  IT IS USUALLY FOLLOWED BY    HES  200
C     CQZVAL  AND POSSIBLY  CQZVEC.                                     HES  210
C                                                                       HES  220
C     ON INPUT-                                                         HES  230
```

```
C                                                                       HES  240
C       NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL          HES  250
C         ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM           HES  260
C         DIMENSION STATEMENT,                                          HES  270
C                                                                       HES  280
C       N IS THE ORDER OF THE MATRICES,                                 HES  290
C                                                                       HES  300
C       A=(AR,AI) CONTAINS A COMPLEX GENERAL MATRIX,                    HES  310
C                                                                       HES  320
C       B=(BR,BI) CONTAINS A COMPLEX GENERAL MATRIX,                    HES  330
C                                                                       HES  340
C       MATZ SHOULD BE SET TO .TRUE. IF THE RIGHT HAND TRANSFORMATIONS  HES  350
C         ARE TO BE ACCUMULATED FOR LATER USE IN COMPUTING              HES  360
C         EIGENVECTORS, AND TO .FALSE. OTHERWISE.                       HES  370
C                                                                       HES  380
C     ON OUTPUT-                                                        HES  390
C                                                                       HES  400
C       A HAS BEEN REDUCED TO UPPER HESSENBERG FORM.  THE ELEMENTS      HES  410
C         BELOW THE FIRST SUBDIAGONAL HAVE BEEN SET TO ZERO, AND THE    HES  420
C         SUBDIAGONAL ELEMENTS HAVE BEEN MADE REAL (AND NON-NEGATIVE),  HES  430
C                                                                       HES  440
C       B HAS BEEN REDUCED TO UPPER TRIANGULAR FORM.  THE ELEMENTS      HES  450
C         BELOW THE MAIN DIAGONAL HAVE BEEN SET TO ZERO,                HES  460
C                                                                       HES  470
C       Z=(ZR,ZI) CONTAINS THE PRODUCT OF THE RIGHT HAND                HES  480
C         TRANSFORMATIONS IF MATZ HAS BEEN SET TO .TRUE.                HES  490
C         OTHERWISE, Z IS NOT REFERENCED.                               HES  500
C                                                                       HES  510
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO B. S. GARBOW,        HES  520
C     APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY         HES  530
C                                                                       HES  540
C     ------------------------------------------------------------------HES  550
C                                                                       HES  560
C     ********** INITIALIZE Z **********                               HES  570
      IF (.NOT. MATZ) GO TO 10                                          HES  580
C                                                                       HES  590
      DO 3 I = 1, N                                                     HES  600
C                                                                       HES  610
         DO 2 J = 1, N                                                  HES  620
            ZR(I,J) = 0.0                                               HES  630
            ZI(I,J) = 0.0                                               HES  640
    2    CONTINUE                                                       HES  650
C                                                                       HES  660
         ZR(I,I) = 1.0                                                  HES  670
    3 CONTINUE                                                          HES  680
C     ********** REDUCE B TO UPPER TRIANGULAR FORM WITH                 HES  690
C                TEMPORARILY REAL DIAGONAL ELEMENTS **********          HES  700
   10 IF (N .LE. 1) GO TO 170                                           HES  710
      NM1 = N - 1                                                       HES  720
C                                                                       HES  730
      DO 100 L = 1, NM1                                                 HES  740
         L1 = L + 1                                                     HES  750
         S = 0.0                                                        HES  760
C                                                                       HES  770
         DO 20 I = L, N                                                 HES  780
            S = S + ABS(BR(I,L)) + ABS(BI(I,L))                         HES  790
   20    CONTINUE                                                       HES  800
C                                                                       HES  810
         IF (S .EQ. 0.0) GO TO 100                                      HES  820
         RHO = 0.0                                                      HES  830
C                                                                       HES  840
         DO 25 I = L, N                                                 HES  850
            BR(I,L) = BR(I,L) / S                                       HES  860
            BI(I,L) = BI(I,L) / S                                       HES  870
            RHO = RHO + BR(I,L)**2 + BI(I,L)**2                         HES  880
   25    CONTINUE                                                       HES  890
C                                                                       HES  900
         R = SQRT(RHO)                                                  HES  910
         XR = CABS(CMPLX(BR(L,L),BI(L,L)))                              HES  920
         IF (XR .EQ. 0.0) GO TO 27                                      HES  930
         RHO = RHO + XR * R                                             HES  940
         U1 = -BR(L,L) / XR                                             HES  950
         U1I = -BI(L,L) / XR                                            HES  960
         YR = R / XR + 1.0                                              HES  970
         BR(L,L) = YR * BR(L,L)                                         HES  980
         BI(L,L) = YR * BI(L,L)                                         HES  990
```

```
      GO TO 28                                                          HES 1000
C                                                                       HES 1010
   27 BR(L,L) = R                                                       HES 1020
      U1 = -1.0                                                         HES 1030
      U1I = 0.0                                                         HES 1040
C                                                                       HES 1050
   28 DO 50 J = L1, N                                                   HES 1060
         T = 0.0                                                        HES 1070
         TI = 0.0                                                       HES 1080
C                                                                       HES 1090
         DO 30 I = L, N                                                 HES 1100
            T = T + BR(I,L) * BR(I,J) + BI(I,L) * BI(I,J)               HES 1110
            TI = TI + BR(I,L) * BI(I,J) - BI(I,L) * BR(I,J)             HES 1120
   30    CONTINUE                                                       HES 1130
C                                                                       HES 1140
         T = T / RHO                                                    HES 1150
         TI = TI / RHO                                                  HES 1160
C                                                                       HES 1170
         DO 40 I = L, N                                                 HES 1180
            BR(I,J) = BR(I,J) - T * BR(I,L) + TI * BI(I,L)              HES 1190
            BI(I,J) = BI(I,J) - T * BI(I,L) - TI * BR(I,L)              HES 1200
   40    CONTINUE                                                       HES 1210
C                                                                       HES 1220
         XI = U1 * BI(L,J) - U1I * BR(L,J)                              HES 1230
         BR(L,J) = U1 * BR(L,J) + U1I * BI(L,J)                         HES 1240
         BI(L,J) = XI                                                   HES 1250
   50 CONTINUE                                                          HES 1260
C                                                                       HES 1270
      DO 80 J = 1, N                                                    HES 1280
         T = 0.0                                                        HES 1290
         TI = 0.0                                                       HES 1300
C                                                                       HES 1310
         DO 60 I = L, N                                                 HES 1320
            T = T + BR(I,L) * AR(I,J) + BI(I,L) * AI(I,J)               HES 1330
            TI = TI + BR(I,L) * AI(I,J) - BI(I,L) * AR(I,J)             HES 1340
   60    CONTINUE                                                       HES 1350
C                                                                       HES 1360
         T = T / RHO                                                    HES 1370
         TI = TI / RHO                                                  HES 1380
C                                                                       HES 1390
         DO 70 I = L, N                                                 HES 1400
            AR(I,J) = AR(I,J) - T * BR(I,L) + TI * BI(I,L)              HES 1410
            AI(I,J) = AI(I,J) - T * BI(I,L) - TI * BR(I,L)              HES 1420
   70    CONTINUE                                                       HES 1430
C                                                                       HES 1440
         XI = U1 * AI(L,J) - U1I * AR(L,J)                              HES 1450
         AR(L,J) = U1 * AR(L,J) + U1I * AI(L,J)                         HES 1460
         AI(L,J) = XI                                                   HES 1470
   80 CONTINUE                                                          HES 1480
C                                                                       HES 1490
      BR(L,L) = R * S                                                   HES 1500
      BI(L,L) = 0.0                                                     HES 1510
C                                                                       HES 1520
      DO 90 I = L1, N                                                   HES 1530
         BR(I,L) = 0.0                                                  HES 1540
         BI(I,L) = 0.0                                                  HES 1550
   90 CONTINUE                                                          HES 1560
C                                                                       HES 1570
  100 CONTINUE                                                          HES 1580
C     ********** REDUCE A TO UPPER HESSENBERG FORM WITH REAL SUBDIAGONALHES 1590
C                ELEMENTS, WHILE KEEPING B TRIANGULAR **********       HES 1600
      DO 160 K = 1, NM1                                                 HES 1610
         K1 = K + 1                                                     HES 1620
C     ********** SET BOTTOM ELEMENT IN K-TH COLUMN OF A REAL ********** HES 1630
         IF (AI(N,K) .EQ. 0.0) GO TO 105                                HES 1640
         R = CABS(CMPLX(AR(N,K),AI(N,K)))                               HES 1650
         U1 = AR(N,K) / R                                               HES 1660
         U1I = AI(N,K) / R                                              HES 1670
         AR(N,K) = R                                                    HES 1680
         AI(N,K) = 0.0                                                  HES 1690
C                                                                       HES 1700
         DO 103 J = K1, N                                               HES 1710
            XI = U1 * AI(N,J) - U1I * AR(N,J)                           HES 1720
            AR(N,J) = U1 * AR(N,J) + U1I * AI(N,J)                      HES 1730
            AI(N,J) = XI                                                HES 1740
  103    CONTINUE                                                       HES 1750
```

```
C                                                                       HES 1760
      XI = U1 * BI(N,N) - U1I * BR(N,N)                                 HES 1770
      BR(N,N) = U1 * BR(N,N) + U1I * BI(N,N)                            HES 1780
      BI(N,N) = XI                                                      HES 1790
  105 IF (K .EQ. NM1) GO TO 170                                         HES 1800
      NK1 = NM1 - K                                                     HES 1810
C     ********** FOR L=N-1 STEP -1 UNTIL K+1 DO -- **********          HES 1820
      DO 150 LB = 1, NK1                                                HES 1830
         L = N - LB                                                     HES 1840
         L1 = L + 1                                                     HES 1850
C     ********** ZERO A(L+1,K) **********                              HES 1860
         S = ABS(AR(L,K)) + ABS(AI(L,K)) + AR(L1,K)                     HES 1870
         IF (S .EQ. 0.0) GO TO 150                                      HES 1880
         U1 = AR(L,K) / S                                               HES 1890
         U1I = AI(L,K) / S                                              HES 1900
         U2 = AR(L1,K) / S                                              HES 1910
         R = SQRT(U1*U1+U1I*U1I+U2*U2)                                  HES 1920
         U1 = U1 / R                                                    HES 1930
         U1I = U1I / R                                                  HES 1940
         U2 = U2 / R                                                    HES 1950
         AR(L,K) = R * S                                                HES 1960
         AI(L,K) = 0.0                                                  HES 1970
         AR(L1,K) = 0.0                                                 HES 1980
C                                                                       HES 1990
         DO 110 J = K1, N                                               HES 2000
            XR = AR(L,J)                                                HES 2010
            XI = AI(L,J)                                                HES 2020
            YR = AR(L1,J)                                               HES 2030
            YI = AI(L1,J)                                               HES 2040
            AR(L,J) = U1 * XR + U1I * XI + U2 * YR                      HES 2050
            AI(L,J) = U1 * XI - U1I * XR + U2 * YI                      HES 2060
            AR(L1,J) = U1 * YR - U1I * YI - U2 * XR                     HES 2070
            AI(L1,J) = U1 * YI + U1I * YR - U2 * XI                     HES 2080
  110    CONTINUE                                                       HES 2090
C                                                                       HES 2100
         XR = BR(L,L)                                                   HES 2110
         BR(L,L) = U1 * XR                                              HES 2120
         BI(L,L) = -U1I * XR                                            HES 2130
         BR(L1,L) = -U2 * XR                                            HES 2140
C                                                                       HES 2150
         DO 120 J = L1, N                                               HES 2160
            XR = BR(L,J)                                                HES 2170
            XI = BI(L,J)                                                HES 2180
            YR = BR(L1,J)                                               HES 2190
            YI = BI(L1,J)                                               HES 2200
            BR(L,J) = U1 * XR + U1I * XI + U2 * YR                      HES 2210
            BI(L,J) = U1 * XI - U1I * XR + U2 * YI                      HES 2220
            BR(L1,J) = U1 * YR - U1I * YI - U2 * XR                     HES 2230
            BI(L1,J) = U1 * YI + U1I * YR - U2 * XI                     HES 2240
  120    CONTINUE                                                       HES 2250
C     ********** ZERO B(L+1,L) **********                              HES 2260
         S = ABS(BR(L1,L1)) + ABS(BI(L1,L1)) + ABS(BR(L1,L))            HES 2270
         IF (S .EQ. 0.0) GO TO 150                                      HES 2280
         U1 = BR(L1,L1) / S                                             HES 2290
         U1I = BI(L1,L1) / S                                            HES 2300
         U2 = BR(L1,L) / S                                              HES 2310
         R = SQRT(U1*U1+U1I*U1I+U2*U2)                                  HES 2320
         U1 = U1 / R                                                    HES 2330
         U1I = U1I / R                                                  HES 2340
         U2 = U2 / R                                                    HES 2350
         BR(L1,L1) = R * S                                              HES 2360
         BI(L1,L1) = 0.0                                                HES 2370
         BR(L1,L) = 0.0                                                 HES 2380
C                                                                       HES 2390
         DO 130 I = 1, L                                                HES 2400
            XR = BR(I,L1)                                               HES 2410
            XI = BI(I,L1)                                               HES 2420
            YR = BR(I,L)                                                HES 2430
            YI = BI(I,L)                                                HES 2440
            BR(I,L1) = U1 * XR + U1I * XI + U2 * YR                     HES 2450
            BI(I,L1) = U1 * XI - U1I * XR + U2 * YI                     HES 2460
            BR(I,L) = U1 * YR - U1I * YI - U2 * XR                      HES 2470
            BI(I,L) = U1 * YI + U1I * YR - U2 * XI                      HES 2480
  130    CONTINUE                                                       HES 2490
C                                                                       HES 2500
         DO 140 I = 1, N                                                HES 2510
```

```
                  XR = AR(I,L1)                                          HES 2520
                  XI = AI(I,L1)                                          HES 2530
                  YR = AR(I,L)                                           HES 2540
                  YI = AI(I,L)                                           HES 2550
                  AR(I,L1) = U1 * XR + U1I * XI + U2 * YR                HES 2560
                  AI(I,L1) = U1 * XI - U1I * XR + U2 * YI                HES 2570
                  AR(I,L) = U1 * YR - U1I * YI - U2 * XR                 HES 2580
                  AI(I,L) = U1 * YI + U1I * YR - U2 * XI                 HES 2590
  140          CONTINUE                                                  HES 2600
C                                                                        HES 2610
               IF (.NOT. MATZ) GO TO 150                                 HES 2620
C                                                                        HES 2630
               DO 145 I = 1, N                                           HES 2640
                  XR = ZR(I,L1)                                          HES 2650
                  XI = ZI(I,L1)                                          HES 2660
                  YR = ZR(I,L)                                           HES 2670
                  YI = ZI(I,L)                                           HES 2680
                  ZR(I,L1) = U1 * XR + U1I * XI + U2 * YR                HES 2690
                  ZI(I,L1) = U1 * XI - U1I * XR + U2 * YI                HES 2700
                  ZR(I,L) = U1 * YR - U1I * YI - U2 * XR                 HES 2710
                  ZI(I,L) = U1 * YI + U1I * YR - U2 * XI                 HES 2720
  145          CONTINUE                                                  HES 2730
C                                                                        HES 2740
  150       CONTINUE                                                     HES 2750
C                                                                        HES 2760
  160    CONTINUE                                                        HES 2770
C                                                                        HES 2780
  170    RETURN                                                          HES 2790
C        ********** LAST CARD OF CQZHES **********                       HES 2800
         END                                                             HES 2810


C                                                                        VAL  10
C        ---------------------------------------------------------------VAL  20
C                                                                        VAL  30
         SUBROUTINE CQZVAL(NM,N,AR,AI,BR,BI,EPS1,ALFR,ALFI,BETA,         VAL  40
        X                                   MATZ,ZR,ZI,IERR)             VAL  50
C                                                                        VAL  60
         INTEGER I,J,K,L,N,EN,K1,K2,LL,L1,NA,NM,ITS,KM1,LM1,             VAL  70
        X        ENM2,IERR,LOR1,ENORN                                     VAL  80
         REAL AR(NM,N),AI(NM,N),BR(NM,N),BI(NM,N),ALFR(N),ALFI(N),       VAL  90
        X       BETA(N),ZR(NM,N),ZI(NM,N)                                VAL 100
         REAL R,S,A1,A2,EP,SH,U1,U2,XI,XR,YI,YR,ANI,A1I,A33,A34,A43,A44, VAL 110
        X       BNI,B11,B33,B44,SHI,U1I,A33I,A34I,A43I,A44I,B33I,B44I,   VAL 120
        X       EPSA,EPSB,EPS1,ANORM,BNORM,B3344,B3344I                  VAL 130
         REAL SQRT,CABS,ABS                                              VAL 140
         INTEGER MAX0                                                    VAL 150
         LOGICAL MATZ                                                    VAL 160
         COMPLEX Z3                                                      VAL 170
         COMPLEX CSQRT,CMPLX                                             VAL 180
         REAL REAL,AIMAG                                                 VAL 190
C                                                                        VAL 200
C                                                                        VAL 210
C                                                                        VAL 220
C                                                                        VAL 230
C                                                                        VAL 240
C        THIS SUBROUTINE IS A COMPLEX ANALOGUE OF STEPS 2 AND 3 OF THE   VAL 250
C        QZ ALGORITHM FOR SOLVING GENERALIZED MATRIX EIGENVALUE PROBLEMS, VAL 260
C        SIAM J. NUMER. ANAL. 10, 241-256(1973) BY MOLER AND STEWART,    VAL 270
C        AS MODIFIED IN TECHNICAL NOTE NASA TN E-7305(1973) BY WARD.     VAL 280
C                                                                        VAL 290
C        THIS SUBROUTINE ACCEPTS A PAIR OF COMPLEX MATRICES, ONE OF THEM VAL 300
C        IN UPPER HESSENBERG FORM AND THE OTHER IN UPPER TRIANGULAR FORM, VAL 310
C        THE HESSENBERG MATRIX MUST FURTHER HAVE REAL SUBDIAGONAL ELEMENTS.VAL 320
C        IT REDUCES THE HESSENBERG MATRIX TO TRIANGULAR FORM USING       VAL 330
C        UNITARY TRANSFORMATIONS WHILE MAINTAINING THE TRIANGULAR FORM   VAL 340
C        OF THE OTHER MATRIX AND FURTHER MAKING ITS DIAGONAL ELEMENTS    VAL 350
C        REAL AND NON-NEGATIVE.  IT THEN RETURNS QUANTITIES WHOSE RATIOS VAL 360
C        GIVE THE GENERALIZED EIGENVALUES.  IT IS USUALLY PRECEDED BY    VAL 370
C        CQZHES  AND POSSIBLY FOLLOWED BY  CQZVEC.                       VAL 380
C                                                                        VAL 390
C        ON INPUT-                                                       VAL 400
C                                                                        VAL 410
C           NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL       VAL 420
C             ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM        VAL 430
```

```
C          DIMENSION STATEMENT,                                          VAL  440
C                                                                        VAL  450
C        N IS THE ORDER OF THE MATRICES,                                 VAL  460
C                                                                        VAL  470
C        A=(AR,AI) CONTAINS A COMPLEX UPPER HESSENBERG MATRIX            VAL  480
C          WITH REAL SUBDIAGONAL ELEMENTS,                               VAL  490
C                                                                        VAL  500
C        B=(BR,BI) CONTAINS A COMPLEX UPPER TRIANGULAR MATRIX,           VAL  510
C                                                                        VAL  520
C        EPS1 IS A TOLERANCE USED TO DETERMINE NEGLIGIBLE ELEMENTS.      VAL  530
C          EPS1 = 0.0 (OR NEGATIVE) MAY BE INPUT, IN WHICH CASE AN       VAL  540
C          ELEMENT WILL BE NEGLECTED ONLY IF IT IS LESS THAN ROUNDOFF    VAL  550
C          ERROR TIMES THE NORM OF ITS MATRIX.  IF THE INPUT EPS1 IS     VAL  560
C          POSITIVE, THEN AN ELEMENT WILL BE CONSIDERED NEGLIGIBLE       VAL  570
C          IF IT IS LESS THAN EPS1 TIMES THE NORM OF ITS MATRIX.  A      VAL  580
C          POSITIVE VALUE OF EPS1 MAY RESULT IN FASTER EXECUTION,        VAL  590
C          BUT LESS ACCURATE RESULTS,                                    VAL  600
C                                                                        VAL  610
C        MATZ SHOULD BE SET TO .TRUE. IF THE RIGHT HAND TRANSFORMATIONS  VAL  620
C          ARE TO BE ACCUMULATED FOR LATER USE IN COMPUTING              VAL  630
C          EIGENVECTORS, AND TO .FALSE. OTHERWISE,                       VAL  640
C                                                                        VAL  650
C        Z=(ZR,ZI) CONTAINS, IF MATZ HAS BEEN SET TO .TRUE., THE         VAL  660
C          TRANSFORMATION MATRIX PRODUCED IN THE REDUCTION               VAL  670
C          BY  CQZHES, IF PERFORMED, OR ELSE THE IDENTITY MATRIX.        VAL  680
C          IF MATZ HAS BEEN SET TO .FALSE., Z IS NOT REFERENCED.         VAL  690
C                                                                        VAL  700
C     ON OUTPUT-                                                         VAL  710
C                                                                        VAL  720
C        A HAS BEEN REDUCED TO UPPER TRIANGULAR FORM.  THE ELEMENTS      VAL  730
C          BELOW THE MAIN DIAGONAL HAVE BEEN SET TO ZERO,                VAL  740
C                                                                        VAL  750
C        B IS STILL IN UPPER TRIANGULAR FORM, ALTHOUGH ITS ELEMENTS      VAL  760
C          HAVE BEEN ALTERED.  IN PARTICULAR, ITS DIAGONAL HAS BEEN SET  VAL  770
C          REAL AND NON-NEGATIVE.  THE LOCATION BR(N,1) IS USED TO       VAL  780
C          STORE EPS1 TIMES THE NORM OF B FOR LATER USE BY  CQZVEC,      VAL  790
C                                                                        VAL  800
C        ALFR AND ALFI CONTAIN THE REAL AND IMAGINARY PARTS OF THE       VAL  810
C          DIAGONAL ELEMENTS OF THE TRIANGULARIZED A MATRIX,             VAL  820
C                                                                        VAL  830
C        BETA CONTAINS THE REAL NON-NEGATIVE DIAGONAL ELEMENTS OF THE    VAL  840
C          CORRESPONDING B.  THE GENERALIZED EIGENVALUES ARE THEN        VAL  850
C          THE RATIOS ((ALFR+I*ALFI)/BETA),                              VAL  860
C                                                                        VAL  870
C        Z CONTAINS THE PRODUCT OF THE RIGHT HAND TRANSFORMATIONS        VAL  880
C          (FOR BOTH STEPS) IF MATZ HAS BEEN SET TO .TRUE.,              VAL  890
C                                                                        VAL  900
C        IERR IS SET TO                                                  VAL  910
C          ZERO       FOR NORMAL RETURN,                                VAL  920
C          J          IF AR(J,J-1) HAS NOT BECOME                       VAL  930
C                     ZERO AFTER 50 ITERATIONS.                         VAL  940
C                                                                        VAL  950
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO B. S. GARBOW,         VAL  960
C     APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY          VAL  970
C                                                                        VAL  980
C     ------------------------------------------------------------------VAL  990
C                                                                        VAL 1000
      IERR = 0                                                           VAL 1010
C     ********** COMPUTE EPSA,EPSB **********                           VAL 1020
      ANORM = 0.0                                                        VAL 1030
      BNORM = 0.0                                                        VAL 1040
C                                                                        VAL 1050
      DO 30 I = 1, N                                                     VAL 1060
         ANI = 0.0                                                       VAL 1070
         IF (I .NE. 1) ANI = ABS(AR(I,I-1))                              VAL 1080
         BNI = 0.0                                                       VAL 1090
C                                                                        VAL 1100
         DO 20 J = I, N                                                  VAL 1110
            ANI = ANI + ABS(AR(I,J)) + ABS(AI(I,J))                      VAL 1120
            BNI = BNI + ABS(BR(I,J)) + ABS(BI(I,J))                      VAL 1130
   20    CONTINUE                                                        VAL 1140
C                                                                        VAL 1150
         IF (ANI .GT. ANORM) ANORM = ANI                                 VAL 1160
         IF (BNI .GT. BNORM) BNORM = BNI                                 VAL 1170
   30 CONTINUE                                                           VAL 1180
C                                                                        VAL 1190
```

```
      IF (ANORM .EQ. 0.0) ANORM = 1.0                                   VAL 1200
      IF (BNORM .EQ. 0.0) BNORM = 1.0                                   VAL 1210
      EP = EPS1                                                         VAL 1220
      IF (EP .GT. 0.0) GO TO 50                                         VAL 1230
C     ********** COMPUTE ROUNDOFF LEVEL IF EPS1 IS ZERO **********     VAL 1240
      EP = 1.0                                                          VAL 1250
   40 EP = EP / 2.0                                                     VAL 1260
      IF (1.0 + EP .GT. 1.0) GO TO 40                                   VAL 1270
   50 EPSA = EP * ANORM                                                 VAL 1280
      EPSB = EP * BNORM                                                 VAL 1290
C     ********** REDUCE A TO TRIANGULAR FORM, WHILE                    VAL 1300
C                KEEPING B TRIANGULAR **********                      VAL 1310
      LOR1 = 1                                                          VAL 1320
      ENORN = N                                                         VAL 1330
      EN = N                                                            VAL 1340
C     ********** BEGIN QZ STEP **********                              VAL 1350
   60 IF (EN .EQ. 0) GO TO 1001                                         VAL 1360
      IF (.NOT. MATZ) ENORN = EN                                        VAL 1370
      ITS = 0                                                           VAL 1380
      NA = EN - 1                                                       VAL 1390
      ENM2 = NA - 1                                                     VAL 1400
C     ********** CHECK FOR CONVERGENCE OR REDUCIBILITY.                 VAL 1410
C                FOR L=EN STEP -1 UNTIL 1 DO -- **********            VAL 1420
   70 DO 80 LL = 1, EN                                                  VAL 1430
         LM1 = EN - LL                                                  VAL 1440
         L = LM1 + 1                                                    VAL 1450
         IF (L .EQ. 1) GO TO 95                                         VAL 1460
         IF (ABS(AR(L,LM1)) .LE. EPSA) GO TO 90                         VAL 1470
   80 CONTINUE                                                          VAL 1480
C                                                                       VAL 1490
   90 AR(L,LM1) = 0.0                                                   VAL 1500
C     ********** SET DIAGONAL ELEMENT AT TOP OF B REAL **********      VAL 1510
   95 B11 = CABS(CMPLX(BR(L,L),BI(L,L)))                                VAL 1520
      IF (BI(L,L) .EQ. 0.0) GO TO 98                                    VAL 1530
      U1 = BR(L,L) / B11                                                VAL 1540
      U1I = BI(L,L) / B11                                               VAL 1550
C                                                                       VAL 1560
      DO 97 J = L, ENORN                                                VAL 1570
         XI = U1 * AI(L,J) - U1I * AR(L,J)                              VAL 1580
         AR(L,J) = U1 * AR(L,J) + U1I * AI(L,J)                         VAL 1590
         AI(L,J) = XI                                                   VAL 1600
         XI = U1 * BI(L,J) - U1I * BR(L,J)                              VAL 1610
         BR(L,J) = U1 * BR(L,J) + U1I * BI(L,J)                         VAL 1620
         BI(L,J) = XI                                                   VAL 1630
   97 CONTINUE                                                          VAL 1640
C                                                                       VAL 1650
      BI(L,L) = 0.0                                                     VAL 1660
   98 IF (L .NE. EN) GO TO 100                                          VAL 1670
C     ********** 1-BY-1 BLOCK ISOLATED **********                      VAL 1680
      ALFR(EN) = AR(EN,EN)                                              VAL 1690
      ALFI(EN) = AI(EN,EN)                                              VAL 1700
      BETA(EN) = B11                                                    VAL 1710
      EN = NA                                                           VAL 1720
      GO TO 60                                                          VAL 1730
C     ********** CHECK FOR SMALL TOP OF B **********                   VAL 1740
  100 L1 = L + 1                                                        VAL 1750
      IF (B11 .GT. EPSB) GO TO 120                                      VAL 1760
      BR(L,L) = 0.0                                                     VAL 1770
      S = ABS(AR(L,L)) + ABS(AI(L,L)) + ABS(AR(L1,L))                   VAL 1780
      U1 = AR(L,L) / S                                                  VAL 1790
      U1I = AI(L,L) / S                                                 VAL 1800
      U2 = AR(L1,L) / S                                                 VAL 1810
      R = SQRT(U1*U1+U1I*U1I+U2*U2)                                     VAL 1820
      U1 = U1 / R                                                       VAL 1830
      U1I = U1I / R                                                     VAL 1840
      U2 = U2 / R                                                       VAL 1850
      AR(L,L) = R * S                                                   VAL 1860
      AI(L,L) = 0.0                                                     VAL 1870
C                                                                       VAL 1880
      DO 110 J = L1, ENORN                                              VAL 1890
         XR = AR(L,J)                                                   VAL 1900
         XI = AI(L,J)                                                   VAL 1910
         YR = AR(L1,J)                                                  VAL 1920
         YI = AI(L1,J)                                                  VAL 1930
         AR(L,J) = U1 * XR + U1I * XI + U2 * YR                         VAL 1940
         AI(L,J) = U1 * XI - U1I * XR + U2 * YI                         VAL 1950
```

```
            AR(L1,J) = U1 * YR - U1I * YI - U2 * XR                    VAL 1960
            AI(L1,J) = U1 * YI + U1I * YR - U2 * XI                    VAL 1970
            XR = BR(L,J)                                               VAL 1980
            XI = BI(L,J)                                               VAL 1990
            YR = BR(L1,J)                                              VAL 2000
            YI = BI(L1,J)                                              VAL 2010
            BR(L1,J) = U1 * YR - U1I * YI - U2 * XR                    VAL 2040
            BR(L,J) = U1 * XR + U1I * XI + U2 * YR                     VAL 2020
            BI(L,J) = U1 * XI - U1I * XR + U2 * YI                     VAL 2030
            BI(L1,J) = U1 * YI + U1I * YR - U2 * XI                    VAL 2050
  110 CONTINUE                                                         VAL 2060
C                                                                      VAL 2070
      LM1 = L                                                          VAL 2080
      L = L1                                                           VAL 2090
      GO TO 90                                                         VAL 2100
C     ********** ITERATION STRATEGY **********                        VAL 2110
  120 IF (ITS .EQ. 50) GO TO 1000                                      VAL 2120
      IF (ITS .EQ. 10) GO TO 135                                       VAL 2130
C     ********** DETERMINE SHIFT **********                           VAL 2140
      B33 = BR(NA,NA)                                                  VAL 2150
      B33I = BI(NA,NA)                                                 VAL 2160
      IF (CABS(CMPLX(B33,B33I)) .GE. EPSB) GO TO 122                   VAL 2170
      B33 = EPSB                                                       VAL 2180
      B33I = 0.0                                                       VAL 2190
  122 B44 = BR(EN,EN)                                                  VAL 2200
      B44I = BI(EN,EN)                                                 VAL 2210
      IF (CABS(CMPLX(B44,B44I)) .GE. EPSB) GO TO 124                   VAL 2220
      B44 = EPSB                                                       VAL 2230
      B44I = 0.0                                                       VAL 2240
  124 B3344 = B33 * B44 - B33I * B44I                                  VAL 2250
      B3344I = B33 * B44I + B33I * B44                                 VAL 2260
      A33 = AR(NA,NA) * B44 - AI(NA,NA) * B44I                         VAL 2270
      A33I = AR(NA,NA) * B44I + AI(NA,NA) * B44                        VAL 2280
      A34 = AR(NA,EN) * B33 - AI(NA,EN) * B33I                         VAL 2290
     X    - AR(NA,NA) * BR(NA,EN) + AI(NA,NA) * BI(NA,EN)             VAL 2300
      A34I = AR(NA,EN) * B33I + AI(NA,EN) * B33                        VAL 2310
     X     - AR(NA,NA) * BI(NA,EN) - AI(NA,NA) * BR(NA,EN)            VAL 2320
      A43 = AR(EN,NA) * B44                                            VAL 2330
      A43I = AR(EN,NA) * B44I                                          VAL 2340
      A44 = AR(EN,EN) * B33 - AI(EN,EN) * B33I - AR(EN,NA) * BR(NA,EN) VAL 2350
      A44I = AR(EN,EN) * B33I + AI(EN,EN) * B33 - AR(EN,NA) * BI(NA,EN) VAL 2360
      SH = A44                                                         VAL 2370
      SHI = A44I                                                       VAL 2380
      XR = A34 * A43 - A34I * A43I                                     VAL 2390
      XI = A34 * A43I + A34I * A43                                     VAL 2400
      IF (XR .EQ. 0.0 .AND. XI .EQ. 0.0) GO TO 140                     VAL 2410
      YR = (A33 - SH) / 2.0                                            VAL 2420
      YI = (A33I - SHI) / 2.0                                          VAL 2430
      Z3 = CSQRT(CMPLX(YR**2-YI**2+XR,2.0*YR*YI+XI))                   VAL 2440
      U1 = REAL(Z3)                                                    VAL 2450
      U1I = AIMAG(Z3)                                                  VAL 2460
      IF (YR * U1 + YI * U1I .GE. 0.0) GO TO 125                       VAL 2470
      U1 = -U1                                                         VAL 2480
      U1I = -U1I                                                       VAL 2490
  125 Z3 = (CMPLX(SH,SHI) - CMPLX(XR,XI) / CMPLX(YR+U1,YI+U1I))        VAL 2500
     X   / CMPLX(B3344,B3344I)                                         VAL 2510
      SH = REAL(Z3)                                                    VAL 2520
      SHI = AIMAG(Z3)                                                  VAL 2530
      GO TO 140                                                        VAL 2540
C     ********** AD HOC SHIFT **********                              VAL 2550
  135 SH = AR(EN,NA) + AR(NA,ENM2)                                     VAL 2560
      SHI = 0.0                                                        VAL 2570
C     ********** DETERMINE ZEROTH COLUMN OF A **********              VAL 2580
  140 A1 = AR(L,L) / B11 - SH                                          VAL 2590
      A1I = AI(L,L) / B11 - SHI                                        VAL 2600
      A2 = AR(L1,L) / B11                                              VAL 2610
      ITS = ITS + 1                                                    VAL 2620
      IF (.NOT. MATZ) LOR1 = L                                         VAL 2630
C     ********** MAIN LOOP **********                                 VAL 2640
      DO 260 K = L, NA                                                 VAL 2650
         K1 = K + 1                                                    VAL 2660
         K2 = K + 2                                                    VAL 2670
         KM1 = MAX0(K-1,L)                                             VAL 2680
C     ********** ZERO A(K+1,K-1) **********                           VAL 2690
         IF (K .EQ. L) GO TO 170                                       VAL 2700
         A1 = AR(K,KM1)                                                VAL 2710
```

```
      A1I = AI(K,KM1)                                                    VAL 2720
      A2 = AR(K1,KM1)                                                    VAL 2730
  170 S = ABS(A1) + ABS(A1I) + ABS(A2)                                   VAL 2740
      U1 = A1 / S                                                        VAL 2750
      U1I = A1I / S                                                      VAL 2760
      U2 = A2 / S                                                        VAL 2770
      R = SQRT(U1*U1+U1I*U1I+U2*U2)                                      VAL 2780
      U1 = U1 / R                                                        VAL 2790
      U1I = U1I / R                                                      VAL 2800
      U2 = U2 / R                                                        VAL 2810
C                                                                        VAL 2820
      DO 180 J = KM1, ENORN                                              VAL 2830
         XR = AR(K,J)                                                    VAL 2840
         XI = AI(K,J)                                                    VAL 2850
         YR = AR(K1,J)                                                   VAL 2860
         YI = AI(K1,J)                                                   VAL 2870
         AR(K,J) = U1 * XR + U1I * XI + U2 * YR                          VAL 2880
         AI(K,J) = U1 * XI - U1I * XR + U2 * YI                          VAL 2890
         AR(K1,J) = U1 * YR - U1I * YI - U2 * XR                         VAL 2900
         AI(K1,J) = U1 * YI + U1I * YR - U2 * XI                         VAL 2910
         XR = BR(K,J)                                                    VAL 2920
         XI = BI(K,J)                                                    VAL 2930
         YR = BR(K1,J)                                                   VAL 2940
         YI = BI(K1,J)                                                   VAL 2950
         BR(K,J) = U1 * XR + U1I * XI + U2 * YR                          VAL 2960
         BI(K,J) = U1 * XI - U1I * XR + U2 * YI                          VAL 2970
         BR(K1,J) = U1 * YR - U1I * YI - U2 * XR                         VAL 2980
         BI(K1,J) = U1 * YI + U1I * YR - U2 * XI                         VAL 2990
  180 CONTINUE                                                           VAL 3000
C                                                                        VAL 3010
      IF (K .EQ. L) GO TO 240                                            VAL 3020
      AI(K,KM1) = 0.0                                                    VAL 3030
      AR(K1,KM1) = 0.0                                                   VAL 3040
      AI(K1,KM1) = 0.0                                                   VAL 3050
C     ********** ZERO B(K+1,K) **********                                VAL 3060
  240 S = ABS(BR(K1,K1)) + ABS(BI(K1,K1)) + ABS(BR(K1,K))                VAL 3070
      U1 = BR(K1,K1) / S                                                 VAL 3080
      U1I = BI(K1,K1) / S                                                VAL 3090
      U2 = BR(K1,K) / S                                                  VAL 3100
      R = SQRT(U1*U1+U1I*U1I+U2*U2)                                      VAL 3110
      U1 = U1 / R                                                        VAL 3120
      U1I = U1I / R                                                      VAL 3130
      U2 = U2 / R                                                        VAL 3140
      IF (K .EQ. NA) GO TO 245                                           VAL 3150
      XR = AR(K2,K1)                                                     VAL 3160
      AR(K2,K1) = U1 * XR                                                VAL 3170
      AI(K2,K1) = -U1I * XR                                              VAL 3180
      AR(K2,K) = -U2 * XR                                                VAL 3190
C                                                                        VAL 3200
  245 DO 250 I = LOR1, K1                                                VAL 3210
         XR = AR(I,K1)                                                   VAL 3220
         XI = AI(I,K1)                                                   VAL 3230
         YR = AR(I,K)                                                    VAL 3240
         YI = AI(I,K)                                                    VAL 3250
         AR(I,K1) = U1 * XR + U1I * XI + U2 * YR                         VAL 3260
         AI(I,K1) = U1 * XI - U1I * XR + U2 * YI                         VAL 3270
         AR(I,K) = U1 * YR - U1I * YI - U2 * XR                          VAL 3280
         AI(I,K) = U1 * YI + U1I * YR - U2 * XI                          VAL 3290
         XR = BR(I,K1)                                                   VAL 3300
         XI = BI(I,K1)                                                   VAL 3310
         YR = BR(I,K)                                                    VAL 3320
         YI = BI(I,K)                                                    VAL 3330
         BR(I,K1) = U1 * XR + U1I * XI + U2 * YR                         VAL 3340
         BI(I,K1) = U1 * XI - U1I * XR + U2 * YI                         VAL 3350
         BR(I,K) = U1 * YR - U1I * YI - U2 * XR                          VAL 3360
         BI(I,K) = U1 * YI + U1I * YR - U2 * XI                          VAL 3370
  250 CONTINUE                                                           VAL 3380
C                                                                        VAL 3390
      BI(K1,K1) = 0.0                                                    VAL 3400
      BR(K1,K) = 0.0                                                     VAL 3410
      BI(K1,K) = 0.0                                                     VAL 3420
      IF (.NOT. MATZ) GO TO 260                                          VAL 3430
C                                                                        VAL 3440
      DO 255 I = 1, N                                                    VAL 3450
         XR = ZR(I,K1)                                                   VAL 3460
         XI = ZI(I,K1)                                                   VAL 3470
```

```
                YR = ZR(I,K)                                                    VAL 3480
                YI = ZI(I,K)                                                    VAL 3490
                ZR(I,K1) = U1 * XR + U1I * XI + U2 * YR                         VAL 3500
                ZI(I,K1) = U1 * XI - U1I * XR + U2 * YI                         VAL 3510
                ZR(I,K) = U1 * YR - U1I * YI - U2 * XR                          VAL 3520
                ZI(I,K) = U1 * YI + U1I * YR - U2 * XI                          VAL 3530
  255       CONTINUE                                                            VAL 3540
C                                                                               VAL 3550
  260 CONTINUE                                                                  VAL 3560
C     ********** SET LAST A SUBDIAGONAL REAL AND END QZ STEP **********        VAL 3570
      IF (AI(EN,NA) .EQ. 0.0) GO TO 70                                          VAL 3580
      R = CABS(CMPLX(AR(EN,NA),AI(EN,NA)))                                      VAL 3590
      U1 = AR(EN,NA) / R                                                        VAL 3600
      U1I = AI(EN,NA) / R                                                       VAL 3610
      AR(EN,NA) = R                                                             VAL 3620
      AI(EN,NA) = 0.0                                                           VAL 3630
C                                                                               VAL 3640
      DO 270 J = EN, ENORN                                                      VAL 3650
         XI = U1 * AI(EN,J) - U1I * AR(EN,J)                                    VAL 3660
         AR(EN,J) = U1 * AR(EN,J) + U1I * AI(EN,J)                              VAL 3670
         AI(EN,J) = XI                                                          VAL 3680
         XI = U1 * BI(EN,J) - U1I * BR(EN,J)                                    VAL 3690
         BR(EN,J) = U1 * BR(EN,J) + U1I * BI(EN,J)                              VAL 3700
         BI(EN,J) = XI                                                          VAL 3710
  270 CONTINUE                                                                  VAL 3720
C                                                                               VAL 3730
      GO TO 70                                                                  VAL 3740
C     ********** SET ERROR -- BOTTOM SUBDIAGONAL ELEMENT HAS NOT                VAL 3750
C                BECOME NEGLIGIBLE AFTER 50 ITERATIONS **********             VAL 3760
 1000 IERR = EN                                                                 VAL 3770
C     ********** SAVE EPSB FOR USE BY CQZVEC **********                        VAL 3780
 1001 IF (N .GT. 1) BR(N,1) = EPSB                                              VAL 3790
      RETURN                                                                    VAL 3800
C     ********** LAST CARD OF CQZVAL **********                                VAL 3810
      END                                                                       VAL 3820


C                                                                               VEC   10
C     ------------------------------------------------------------------------VEC   20
C                                                                               VEC   30
      SUBROUTINE CQZVEC(NM,N,AR,AI,BR,BI,ALFR,ALFI,BETA,ZR,ZI)                  VEC   40
C                                                                               VEC   50
      INTEGER I,J,K;M,N,EN,II,JJ,NA,NM,NN                                      VEC   60
      REAL AR(NM,N),AI(NM,N),BR(NM,N),BI(NM,N),ALFR(N),ALFI(N),                 VEC   70
     X       BETA(N),ZR(NM,N),ZI(NM,N)                                          VEC   80
      REAL R,T,RI,TI,XI,ALMI,ALMR,BETM,EPSB                                     VEC   90
      REAL CABS                                                                 VEC  100
      COMPLEX Z3                                                                VEC  110
      COMPLEX CMPLX                                                             VEC  120
      REAL REAL,AIMAG                                                           VEC  130
C                                                                               VEC  140
C                                                                               VEC  150
C                                                                               VEC  160
C                                                                               VEC  170
C                                                                               VEC  180
C     THIS SUBROUTINE IS A COMPLEX ANALOGUE OF THE FOURTH STEP OF THE           VEC  190
C     QZ ALGORITHM FOR SOLVING GENERALIZED MATRIX EIGENVALUE PROBLEMS,          VEC  200
C     SIAM J. NUMER. ANAL. 10, 241-256(1973) BY MOLER AND STEWART.              VEC  210
C                                                                               VEC  220
C     THIS SUBROUTINE ACCEPTS A PAIR OF COMPLEX MATRICES IN UPPER               VEC  230
C     TRIANGULAR FORM, WHERE ONE OF THEM FURTHER MUST HAVE REAL DIAGONALVEC  240
C     ELEMENTS.  IT COMPUTES THE EIGENVECTORS OF THE TRIANGULAR PROBLEM VEC  250
C     AND TRANSFORMS THE RESULTS BACK TO THE ORIGINAL COORDINATE SYSTEM.VEC  260
C     IT IS USUALLY PRECEDED BY  CQZHES  AND  CQZVAL.                           VEC  270
C                                                                               VEC  280
C     ON INPUT-                                                                 VEC  290
C                                                                               VEC  300
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL                 VEC  310
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM                  VEC  320
C          DIMENSION STATEMENT,                                                 VEC  330
C                                                                               VEC  340
C        N IS THE ORDER OF THE MATRICES,                                        VEC  350
C                                                                               VEC  360
C        A=(AR,AI) CONTAINS A COMPLEX UPPER TRIANGULAR MATRIX,                  VEC  370
C                                                                               VEC  380
C        B=(BR,BI) CONTAINS A COMPLEX UPPER TRIANGULAR MATRIX WITH REAL         VEC  390
```

```
C          DIAGONAL ELEMENTS.  IN ADDITION, LOCATION BR(N,1) CONTAINS   VEC  400
C          THE TOLERANCE QUANTITY (EPSB) COMPUTED AND SAVED IN  CQZVAL, VEC  410
C                                                                       VEC  420
C        ALFR, ALFI, AND BETA ARE VECTORS WITH COMPONENTS WHOSE         VEC  430
C          RATIOS ((ALFR+I*ALFI)/BETA) ARE THE GENERALIZED              VEC  440
C          EIGENVALUES.  THEY ARE USUALLY OBTAINED FROM  CQZVAL,        VEC  450
C                                                                       VEC  460
C        Z=(ZR,ZI) CONTAINS THE TRANSFORMATION MATRIX PRODUCED IN THE   VEC  470
C          REDUCTIONS BY  CQZHES  AND  CQZVAL, IF PERFORMED.            VEC  480
C          IF THE EIGENVECTORS OF THE TRIANGULAR PROBLEM ARE            VEC  490
C          DESIRED, Z MUST CONTAIN THE IDENTITY MATRIX.                 VEC  500
C                                                                       VEC  510
C     ON OUTPUT-                                                        VEC  520
C                                                                       VEC  530
C        A IS UNALTERED,                                                VEC  540
C                                                                       VEC  550
C        B HAS BEEN DESTROYED,                                          VEC  560
C                                                                       VEC  570
C        ALFR, ALFI, AND BETA ARE UNALTERED,                            VEC  580
C                                                                       VEC  590
C        Z CONTAINS THE EIGENVECTORS.  EACH EIGENVECTOR IS NORMALIZED   VEC  600
C          SO THAT THE MODULUS OF ITS LARGEST COMPONENT IS 1.0 .         VEC  610
C                                                                       VEC  620
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO B. S. GARBOW,        VEC  630
C     APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY         VEC  640
C                                                                       VEC  650
C     ------------------------------------------------------------------VEC  660
C                                                                       VEC  670
      IF (N .LE. 1) GO TO 1001                                          VEC  680
      EPSB = BR(N,1)                                                    VEC  690
C     ********** FOR EN=N STEP -1 UNTIL 2 DO -- **********              VEC  700
      DO 800 NN = 2, N                                                  VEC  710
         EN = N + 2 - NN                                                VEC  720
         NA = EN - 1                                                    VEC  730
         ALMR = ALFR(EN)                                                VEC  740
         ALMI = ALFI(EN)                                                VEC  750
         BETM = BETA(EN)                                                VEC  760
C        ********** FOR I=EN-1 STEP -1 UNTIL 1 DO -- **********         VEC  770
         DO 700 II = 1, NA                                              VEC  780
            I = EN - II                                                 VEC  790
            R = 0.0                                                     VEC  800
            RI = 0.0                                                    VEC  810
            M = I + 1                                                   VEC  820
C                                                                       VEC  830
            DO 610 J = M, EN                                            VEC  840
               T = BETM * AR(I,J) - ALMR * BR(I,J) + ALMI * BI(I,J)     VEC  850
               TI = BETM * AI(I,J) - ALMR * BI(I,J) - ALMI * BR(I,J)    VEC  860
               IF (J .EQ. EN) GO TO 605                                 VEC  870
               XI = T * BI(J,EN) + TI * BR(J,EN)                        VEC  880
               T = T * BR(J,EN) - TI * BI(J,EN)                         VEC  890
               TI = XI                                                  VEC  900
  605          R = R + T                                                VEC  910
               RI = RI + TI                                             VEC  920
  610       CONTINUE                                                    VEC  930
C                                                                       VEC  940
            T = ALMR * BETA(I) - BETM * ALFR(I)                         VEC  950
            TI = ALMI * BETA(I) - BETM * ALFI(I)                        VEC  960
            IF (T .EQ. 0.0 .AND. TI .EQ. 0.0) T = EPSB                  VEC  970
            Z3 = CMPLX(R,RI) / CMPLX(T,TI)                              VEC  980
            BR(I,EN) = REAL(Z3)                                         VEC  990
            BI(I,EN) = AIMAG(Z3)                                        VEC 1000
  700    CONTINUE                                                       VEC 1010
C                                                                       VEC 1020
  800 CONTINUE                                                          VEC 1030
C     ********** END BACK SUBSTITUTION.                                 VEC 1040
C                TRANSFORM TO ORIGINAL COORDINATE SYSTEM.               VEC 1050
C                FOR J=N STEP -1 UNTIL 2 DO -- **********               VEC 1060
      DO 880 JJ = 2, N                                                  VEC 1070
         J = N + 2 - JJ                                                 VEC 1080
         M = J - 1                                                      VEC 1090
C                                                                       VEC 1100
         DO 880 I = 1, N                                                VEC 1110
C                                                                       VEC 1120
            DO 860 K = 1, M                                             VEC 1130
               ZR(I,J) = ZR(I,J) + ZR(I,K) * BR(K,J) - ZI(I,K) * BI(K,J)VEC 1140
               ZI(I,J) = ZI(I,J) + ZR(I,K) * BI(K,J) + ZI(I,K) * BR(K,J)VEC 1150
```

```
  860       CONTINUE                                                    VEC 1160
C                                                                       VEC 1170
  880 CONTINUE                                                          VEC 1180
C     ********** NORMALIZE SO THAT MODULUS OF LARGEST                   VEC 1190
C                COMPONENT OF EACH VECTOR IS 1 **********               VEC 1200
      DO 950 J = 1, N                                                   VEC 1210
         T = 0.0                                                        VEC 1220
C                                                                       VEC 1230
         DO 930 I = 1, N                                                VEC 1240
            R = CABS(CMPLX(ZR(I,J),ZI(I,J)))                            VEC 1250
            IF (R .GT. T) T = R                                         VEC 1260
  930    CONTINUE                                                       VEC 1270
C                                                                       VEC 1280
         DO 940 I = 1, N                                                VEC 1290
            ZR(I,J) = ZR(I,J) / T                                       VEC 1300
            ZI(I,J) = ZI(I,J) / T                                       VEC 1310
  940    CONTINUE                                                       VEC 1320
C                                                                       VEC 1330
  950 CONTINUE                                                          VEC 1340
C                                                                       VEC 1350
 1001 RETURN                                                            VEC 1360
C     ********** LAST CARD OF CQZVEC **********                        VEC 1370
      END                                                               VEC 1380


C                                                                       CGG   10
C     THIS DRIVER TESTS  QZ  FOR THE CLASS OF COMPLEX GENERALIZED MATRIXCGG   20
C     SYSTEMS EXHIBITING THE USE OF  QZ  TO FIND ALL THE EIGENVALUES    CGG   30
C     AND EIGENVECTORS FOR THE EIGENPROBLEM   A*X = (LAMBDA)*B*X .      CGG   40
C                                                                       CGG   50
C     THE DIMENSION OF  A,AI,B,BI,Z,  AND  ZI  SHOULD BE  NM  BY  NM.   CGG   60
C     THE DIMENSION OF  ALFR,ALFI,BETA,  AND  NORM  SHOULD BE  NM.      CGG   70
C     THE DIMENSION OF  AHOLD  AND  BHOLD  SHOULD BE  NM  BY  NM.       CGG   80
C     THE DIMENSION OF  AHOLDI  AND  BHOLDI  SHOULD BE  NM  BY  NM.     CGG   90
C     HERE NM = 20.                                                     CGG  100
C                                                                       CGG  110
      REAL  A( 20, 20),B( 20, 20),Z( 20, 20),                           CGG  120
     $        ALFR( 20),ALFI( 20),BETA( 20),NORM( 20),                  CGG  130
     $        RESDUL,EPS1, AI(20,20),BI(20,20),ZI(20,20),               CGG  140
     $        AHOLD( 20, 20),BHOLD( 20, 20), AHOLDI(20,20),BHOLDI(20,20)CGG  150
      REAL AMAX1                                                        CGG  160
      INTEGER  ERROR                                                    CGG  170
      DATA IWRITE/6/                                                    CGG  180
C                                                                       CGG  190
      NM = 20                                                           CGG  200
   10 CALL  RMATIN(NM,N,A,B,AHOLD,BHOLD,0)                              CGG  210
      CALL  RMATIN(NM,N,AI,BI,AHOLDI,BHOLDI,0)                          CGG  220
      WRITE(IWRITE,20)                                                  CGG  230
     $             N                                                    CGG  240
   20 FORMAT(30H1THE FULL MATRIX  A   OF ORDER,                         CGG  250
     $       I4,22H  IS (PRINTED BY ROWS)/)                             CGG  260
      DO 30 I = 1,N                                                     CGG  270
   30    WRITE(IWRITE,40)                                               CGG  280
     $              (A(I,J),AI(I,J),J=1,N)                             CGG  290
   40    FORMAT(5(2F6.0,1H1,3X))                                        CGG  300
      WRITE(IWRITE,41)                                                  CGG  310
     $             N                                                    CGG  320
   41 FORMAT(30HOTHE FULL MATRIX  B   OF ORDER,                         CGG  330
     $       I4,22H  IS (PRINTED BY ROWS)/)                             CGG  340
      DO 42 I = 1,N                                                     CGG  350
   42    WRITE(IWRITE,43)                                               CGG  360
     $              (B(I,J),BI(I,J),J=1,N)                             CGG  370
   43    FORMAT(5(2F6.0,1HI,3X))                                        CGG  380
C                                                                       CGG  390
C     EIGENVALUES AND EIGENVECTORS USING  CQZVAL  AND  CQZVEC           CGG  400
C                                                                       CGG  410
         EPS1 = 0.0                                                     CGG  420
         CALL  CQZHES(NM,N,A,AI,B,BI,.TRUE.,Z,ZI)                       CGG  430
         CALL  CQZVAL(NM,N,A,AI,B,BI,EPS1,ALFR,ALFI,BETA,               CGG  440
     $                .TRUE.,Z,ZI,ERROR)                                CGG  450
         CALL  CQZVEC(NM,N,A,AI,B,BI,ALFR,ALFI,BETA,Z,ZI)               CGG  460
         WRITE(IWRITE,292)                                              CGG  470
  292    FORMAT(/15X,7HALFR(I),19X,7HALFI(I),19X,7HBETA(I))             CGG  480
         DO 295 I = 1,N                                                 CGG  490
            WRITE(IWRITE,293)                                           CGG  500
     $                  I,ALFR(I),ALFI(I),BETA(I)                      CGG  510
```

```
  293       FORMAT(I2,3(1PE23.6,3X))                                      CGG  520
  295     CONTINUE                                                        CGG  530
          CALL    RMATIN(NM,N,A,B,AHOLD,BHOLD,1)                          CGG  540
          CALL    RMATIN(NM,N,AI,BI,AHOLDI,BHOLDI,1)                      CGG  550
          CALL    RMATIN(NM,N,AHOLD,AHOLDI,Z,ZI,1)                        CGG  560
          CALL    CGGWZR(NM,N,A,AI,B,BI,ALFR,ALFI,BETA,AHOLD,AHOLDI,      CGG  570
     $            NORM,RESDUL)                                           CGG  580
          WRITE(IWRITE,300)                                               CGG  590
  300 FORMAT(/14X,35HCOMPUTED EIGENVALUE AND EIGENVECTOR,20X,8HRESIDUAL)CGG  600
C                                                                         CGG  610
          DO 510 K = 1, N                                                 CGG  620
             BETA(K) = AMAX1(BETA(K),1.0E-50)                             CGG  630
             ALFR(K) = ALFR(K) / BETA(K)                                  CGG  640
             ALFI(K) = ALFI(K) / BETA(K)                                  CGG  650
C                                                                         CGG  660
C         ONE EIGENVECTOR.                                                CGG  670
C                                                                         CGG  680
  340     WRITE(IWRITE,350)                                               CGG  690
     $               K,ALFR(K),ALFI(K),NORM(K),(Z(I,K),                 CGG  700
     $               ZI(I,K),I=1,N)                                     CGG  710
  350     FORMAT(/I2,1P2E23.6,E29.2/(5X,2E23.6))                          CGG  720
  510 CONTINUE                                                            CGG  730
      GO TO  10                                                           CGG  740
      END                                                                 CGG  750
      SUBROUTINE RMATIN(NM,N,A,B,AHOLD,BHOLD,INITIL)                      CGG  760
C                                                                         CGG  770
C     THIS INPUT SUBROUTINE READS TWO REAL MATRICES  A  AND  B  FROM      CGG  780
C     SYSIN OF ORDER N.                                                   CGG  790
C     TO GENERATE THE MATRICES INITIALLY,  INITIL  IS TO BE 0.            CGG  800
C     TO REGENERATE THE MATRICES FOR THE PURPOSE OF THE RESIDUAL          CGG  810
C     CALCULATION,  INITIL  IS TO BE  1.                                  CGG  820
C                                                                         CGG  830
C     THIS ROUTINE IS CATALOGUED AS  EISPDRV4(RSGREADI).                  CGG  840
C                                                                         CGG  850
      REAL  A(NM,NM),B(NM,NM),AHOLD(NM,NM),BHOLD(NM,NM)                   CGG  860
      REAL  FLOAT                                                         CGG  870
      INTEGER  IA( 20), IB( 20)                                           CGG  880
      DATA IREADA/1/,IREADB/2/,IWRITE/6/                                  CGG  890
C                                                                         CGG  900
      IF( INITIL .EQ. 1 )  GO TO  30                                      CGG  910
      READ(IREADA,5)                                                      CGG  920
     $          N, M                                                     CGG  930
    5 FORMAT(I6,6X,I6)                                                    CGG  940
      IF( N .EQ. 0 )  GO TO  70                                           CGG  950
      IF (M .NE. 1) GO TO 16                                              CGG  960
      DO  10  I = 1,N                                                     CGG  970
         READ(IREADA,17)                                                  CGG  980
     $             (IA(J), J=I,N)                                       CGG  990
         DO   9  J = I,N                                                  CGG 1000
           A(I,J) = FLOAT(IA(J))                                          CGG 1010
    9      A(J,I) =  A(I,J)                                              CGG 1020
   10 CONTINUE                                                            CGG 1030
   11 READ(IREADB,5)                                                      CGG 1040
     $          N,M                                                      CGG 1050
      IF( M .NE. 1 ) GO TO 20                                             CGG 1060
      DO 15 I = 1,N                                                       CGG 1070
         READ(IREADB,17)                                                  CGG 1080
     $             (IB(J), J=I,N)                                       CGG 1090
         DO 14 J = I,N                                                    CGG 1100
            B(I,J)=FLOAT(IB(J))                                           CGG 1110
   14       B(J,I)=B(I,J)                                                 CGG 1120
   15 CONTINUE                                                            CGG 1130
      GO TO 22                                                            CGG 1140
   16 DO  18  I = 1,N                                                     CGG 1150
         READ(IREADA,17)                                                  CGG 1160
     $             (IA(J), J=1,N)                                       CGG 1170
   17    FORMAT(6I12)                                                     CGG 1180
         DO  18  J = 1,N                                                  CGG 1190
   18      A(I,J) = FLOAT(IA(J))                                          CGG 1200
      GO TO 11                                                            CGG 1210
   20 DO 25 I = 1,N                                                       CGG 1220
         READ(IREADB,17)                                                  CGG 1230
     $             (IB(J),J=1,N)                                        CGG 1240
         DO 25 J = 1,N                                                    CGG 1250
   25    B(I,J) = FLOAT(IB(J))                                            CGG 1260
   22 DO  23  I = 1,N                                                     CGG 1270
```

```
      DO  23  J = 1,N                                                   CGG 1280
        BHOLD(I,J) = B(I,J)                                             CGG 1290
   23   AHOLD(I,J) = A(I,J)                                             CGG 1300
      RETURN                                                            CGG 1310
   30 DO  40  I = 1,N                                                   CGG 1320
      DO  40  J = 1,N                                                   CGG 1330
        B(I,J) = BHOLD(I,J)                                             CGG 1340
   40   A(I,J) = AHOLD(I,J)                                             CGG 1350
      RETURN                                                            CGG 1360
   70 WRITE(IWRITE,80)                                                  CGG 1370
   80 FORMAT(47HOEND OF DATA FOR SUBROUTINE  RMATIN(RSGREADI). /1H1)    CGG 1380
      STOP                                                              CGG 1390
      END                                                               CGG 1400
C                                                                       CGG 1410
C     ------------------------------------------------------------      CGG 1420
C                                                                       CGG 1430
      SUBROUTINE CGGWZR(NM,N,A,AI,B,BI,ALFR,ALFI,BETA,Z,ZI,NORM,RESDUL) CGG 1440
C                                                                       CGG 1450
      REAL NORM(N), ALFR(N), ALFI(N), BETA(N), CPART(2), A(NM,N),       CGG 1460
     $       B(NM,N), Z(NM,N),  XR, XI, S, SUMZ, SUMR, SUMI,           CGG 1470
     $       RESDUL,NORMAB,SUMA,SUMB,NORMA,NORMB,                       CGG 1480
     $       SUMR2,SUMI2,SUMR3,SUMI3, AI(NM,N),BI(NM,N),ZI(NM,N)        CGG 1490
      REAL  CABS,ABS,FLOAT,AMAX1                                        CGG 1500
      COMPLEX C, C1                                                     CGG 1510
      COMPLEX  CMPLX                                                    CGG 1520
      EQUIVALENCE(C1,CPART(1))                                          CGG 1530
C                                                                       CGG 1540
C     THIS SUBROUTINE FORMS THE 1-NORM OF THE RESIDUAL MATRIX           CGG 1550
C     A*Z-B*Z*DIAG(W) WHERE A AND B ARE COMPLEX GENERAL MATRICES, Z IS  CGG 1560
C     A MATRIX WHICH CONTAINS THE EIGENVECTORS OF THE EIGENPROBLEM      CGG 1570
C     A*Z - B*Z*DIAG(W), AND  W  STANDS FOR A VECTOR OF CORRESPONDING   CGG 1580
C     EIGENVALUES OF THE EIGENPROBLEM OBTAINED FROM THE VECTORS  ALFR,  CGG 1590
C     ALFI,  AND  BETA  BY THE CORRESPONDENCES                          CGG 1600
C                    W(J) = (ALFR(J) + I*ALFI(J)) / BETA(J) .           CGG 1610
C     ALL NORMS APPEARING IN THE COMMENTS BELOW ARE 1-NORMS.            CGG 1620
C                                                                       CGG 1630
C     INPUT-                                                            CGG 1640
C                                                                       CGG 1650
C        NM IS THE ROW DIMENSION OF TWO-DIMENSIONAL ARRAY PARAMETERS    CGG 1660
C           AS DECLARED IN THE CALLING PROGRAM DIMENSION STATEMENT,     CGG 1670
C                                                                       CGG 1680
C        N IS THE ORDER OF THE MATRICES  A  AND  B,                     CGG 1690
C                                                                       CGG 1700
C        A(NM,N),AI(NM,N),B(NM,N), AND BI(NM,N)  ARE ARRAYS WHICH       CGG 1710
C           CONTAIN THE MATRICES OF THE SYSTEM,                         CGG 1720
C                                                                       CGG 1730
C        Z(NM,N) AND ZI(NM,N)  ARE ARRAYS WHICH CONTAIN THE             CGG 1740
C           EIGENVECTORS OF THE SYSTEM,                                 CGG 1750
C                                                                       CGG 1760
C        ALFR(N), ALFI(N), AND BETA(N)  ARE ARRAYS CONTAINING THE       CGG 1770
C           COMPONENTS OF THE EIGENVALUES OF THE SYSTEM.                CGG 1780
C                                                                       CGG 1790
C     OUTPUT-                                                           CGG 1800
C                                                                       CGG 1810
C        Z(NM,N) AND ZI(NM,N)  ARE ARRAYS WHICH CONTAIN THE NORMALIZED  CGG 1820
C           APPROXIMATE EIGENVECTORS OF THE SYSTEM.  THE EIGENVECTORS   CGG 1830
C           ARE NORMALIZED USING THE 1-NORM IN SUCH A WAY THAT THE FIRSTCGG 1840
C           ELEMENT WHOSE MAGNITUDE IS LARGER THAN THE NORM OF THE      CGG 1850
C           EIGENVECTOR DIVIDED BY N IS REAL AND POSITIVE,              CGG 1860
C                                                                       CGG 1870
C        NORM(N) IS AN ARRAY SUCH THAT FOR EACH K,                      CGG 1880
C                                                                       CGG 1890
C                          ..BETA(K)*A*Z(K)-ALFA*B*Z(K)..               CGG 1900
C         NORM(K) = ----------------------------------------------      CGG 1910
C                   ..Z(K)..*(BETA(K)*..A..+.ALFA(K).*..B..)            CGG 1920
C                                                                       CGG 1930
C         WHERE Z(K) IS THE K-TH EIGENVECTOR AND ALFA = (ALFR + I*ALFI),CGG 1940
C                                                                       CGG 1950
C        RESDUL IS THE REAL NUMBER                                      CGG 1960
C           ..BETA*A*Z-ALFA*B*Z../(..Z..*(BETA*..A..+.ALFA.*..B..))     CGG 1970
C                                                                       CGG 1980
C     ------------------------------------------------------------      CGG 1990
C                                                                       CGG 2000
      NORMB = 0.0                                                       CGG 2010
      NORMA = 0.0                                                       CGG 2020
      RESDUL = 0.0                                                      CGG 2030
```

```
C                                                                       CGG 2040
      DO 20 I = 1,N                                                     CGG 2050
        SUMA = 0.0                                                      CGG 2060
        SUMB = 0.0                                                      CGG 2070
        DO 10 L = 1,N                                                   CGG 2080
          SUMA = SUMA + ABS(A(L,I)) + ABS(AI(L,I))                      CGG 2090
   10     SUMB = SUMB + ABS(B(L,I)) + ABS(BI(L,I))                      CGG 2100
        NORMA = AMAX1(NORMA,SUMA)                                       CGG 2110
   20   NORMB = AMAX1(NORMB,SUMB)                                       CGG 2120
C                                                                       CGG 2130
      DO 160 I=1,N                                                      CGG 2140
         S = 0.0                                                        CGG 2150
         SUMZ = 0.0                                                     CGG 2160
         DO 110 L = 1,N                                                 CGG 2170
            SUMR = 0.0                                                  CGG 2180
            SUMI = 0.0                                                  CGG 2190
            SUMR2 = 0.0                                                 CGG 2200
            SUMI2 = 0.0                                                 CGG 2210
            SUMZ = SUMZ + CABS(CMPLX(Z(L,I),ZI(L,I)))                   CGG 2220
C                                                                       CGG 2230
            DO 100 K=1,N                                                CGG 2240
               SUMR = SUMR + B(L,K)*Z(K,I) - BI(L,K)*ZI(K,I)            CGG 2250
               SUMI = SUMI + B(L,K)*ZI(K,I) + BI(L,K)*Z(K,I)            CGG 2260
               SUMR2 = SUMR2 + A(L,K)*Z(K,I) - AI(L,K)*ZI(K,I)          CGG 2270
  100          SUMI2 = SUMI2 + A(L,K)*ZI(K,I) + AI(L,K)*Z(K,I)          CGG 2280
            SUMR3 = -ALFR(I)*SUMR + ALFI(I)*SUMI                        CGG 2290
            SUMI3 = -ALFI(I)*SUMR - ALFR(I)*SUMI                        CGG 2300
C                                                                       CGG 2310
  110       S = S+CABS(CMPLX(SUMR3,SUMI3)+CMPLX(SUMR2,SUMI2)*BETA(I))   CGG 2320
            NORMAB = NORMA*BETA(I)+NORMB*CABS(CMPLX(ALFR(I),ALFI(I)))   CGG 2330
            IF( NORMAB .EQ. 0.0 ) NORMAB = 1.0                          CGG 2340
C                                                                       CGG 2350
         NORM(I) = SUMZ                                                 CGG 2360
         IF( SUMZ .EQ. 0.0 ) GO TO 150                                  CGG 2370
C        **********THIS LOOP WILL NEVER BE COMPLETED SINCE THERE WILL    CGG 2380
C                  ALWAYS EXIST AN ELEMENT IN THE VECTOR (Z(I),ZI(I))    CGG 2390
C                  LARGER THAN ..(Z(I),ZI(I)../N**********              CGG 2400
         DO 120 L=1,N                                                   CGG 2410
            IF(CABS(CMPLX(Z(L,I),ZI(L,I))) .GE. NORM(I)/FLOAT(N))       CGG 2420
     $      GO TO 130                                                   CGG 2430
  120       CONTINUE                                                    CGG 2440
C                                                                       CGG 2450
  130    XR = NORM(I)*Z(L,I)/CABS(CMPLX(Z(L,I),ZI(L,I)))                CGG 2460
         XI = NORM(I)*ZI(L,I)/CABS(CMPLX(Z(L,I),ZI(L,I)))               CGG 2470
         C = CMPLX(XR,XI)                                               CGG 2480
C                                                                       CGG 2490
         DO 140 L= 1,N                                                  CGG 2500
            C1 = CMPLX(Z(L,I),ZI(L,I))/C                                CGG 2510
            Z(L,I) = CPART(1)                                           CGG 2520
  140       ZI(L,I) = CPART(2)                                          CGG 2530
C                                                                       CGG 2540
         NORM(I) = S/(NORM(I)*NORMAB)                                   CGG 2550
  150    RESDUL = AMAX1(NORM(I),RESDUL)                                 CGG 2560
  160    CONTINUE                                                       CGG 2570
C                                                                       CGG 2580
      RETURN                                                            CGG 2590
      END                                                               CGG 2600
```

```
    5                                                                   DATA1  1
   -238          86         164        -166          56                DATA1  2
     76         -96          40          60         -60                DATA1  3
    118          55         -13          34        -176                DATA1  4
   -314         132         114         -90        -424                DATA1  5
    -54        -205         109         158         -38                DATA1  6
    5                                                                   DATA1  7
   -344         178         240        -308         158                DATA1  8
    152        -128         -32         184        -136                DATA1  9
    284        -182         460        -192        -214                DATA1 10
   -160          78         296        -164        -374                DATA1 11
    -24        -400         148         312         -96                DATA1 12
    5                                                                   DATA1 13
     41        -143         -20          20         104                DATA1 14
    148         144          -6         -78           8                DATA1 15
    -19          87           4         -56        -164                DATA1 16
    -60         -81          99          34          84                DATA1 17
      1         133         132         -46         -12                DATA1 18
    5                                                                   DATA1 19
```

```
   -369    -747   -1368     486    -432          DATA1 20
    261     666   -1152      45    -540          DATA1 21
    819     243    1548    -954     180          DATA1 22
   -945    -279     171     441    -144          DATA1 23
   -468     747     774     -45    -216          DATA1 24
5                                                DATA1 25
    -15    -143     -83      41      55          DATA1 26
    100     144     -60     -60     -34          DATA1 27
    -19      87       4     -56    -164          DATA1 28
   -116     -81      36      55      35          DATA1 29
    -39     133      87     -31     -47          DATA1 30
5                                                DATA1 31
  -1635    -173   -1142    1012    -566          DATA1 32
   -829    1128   -1050     495    -710          DATA1 33
    971     187    1558   -1016     218          DATA1 34
   -562    -944   -1310     238    -937          DATA1 35
    356    -149    -875    -434   -1015          DATA1 36
0                                                DATA1 37
5                                                DATA2  1
    388    -386    -250     556    -396          DATA2  2
   -304     384    -160    -240     240          DATA2  3
   -658     -73    -109    -118     406          DATA2  4
   -640     204    -692     288    -192          DATA2  5
   -162     631     131    -758     278          DATA2  6
5                                                DATA2  7
     94    -122     -14     130     -62          DATA2  8
    -76      64      16     -92      68          DATA2  9
   -136     100    -250     100      96          DATA2 10
    -10     -42     -90      66     154          DATA2 11
    -72     158      52    -184      76          DATA2 12
5                                                DATA2 13
     90     180      36     -90     -72          DATA2 14
   -105    -210     -42     105      84          DATA2 15
    -90    -180     -36      90      72          DATA2 16
     75     150      30     -75     -60          DATA2 17
    -75    -150     -30      75      60          DATA2 18
5                                                DATA2 19
    161     335     182    -162     -36          DATA2 20
   -169    -322      24     167     204          DATA2 21
   -211    -307    -160     186      36          DATA2 22
    205     215      45    -165     -80          DATA2 23
    -48    -299    -102      89      88          DATA2 24
5                                                DATA2 25
     90     180      36     -90     -72          DATA2 26
   -105    -210     -42     105      84          DATA2 27
    -90    -180     -36      90      72          DATA2 28
     75     150      30     -75     -60          DATA2 29
    -75    -150     -30      75      60          DATA2 30
5                                                DATA2 31
    307     253     163    -235     -39          DATA2 32
    -49    -410      46      85     182          DATA2 33
   -229    -253    -200     234      84          DATA2 34
    171     301     184    -130      25          DATA2 35
   -134    -185      59     146     195          DATA2 36
```


Remark on Algorithm 535

The QZ Algorithm to Solve the Generalized Eigenvalue Problem [B.S. Garbow, *ACM Trans. Math. Softw. 4*, 4 (Dec. 1978), 404–410]

B.S. Garbow [Received 10 June 1982; accepted 10 June 1982]

Applied Mathematics Division, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439.

Replace the single card with sequence label **VAL 153**:

IF (BI (L, L) .EQ. 0.0) GO TO 98

with

IF (B11 .EQ. 0.0) GO TO 98

The error that this change corrects results occasionally in eigenvalues with the wrong sign.

ALGORITHM 536
An Efficient One-Way Enciphering Algorithm [Z]

H. D. KNOBLE
The Pennsylvania State University

Key Words and Phrases: one-way security transformation, password, encipher, decipher, multiprecision integer arithmetic
CR Categories: 3.15, 4.49
Language: Fortran

DESCRIPTION

This algorithm is a Fortran implementation of the procedures developed in [1]. Its purpose is to serve as a machine independent model for studying the evaluation of polynomials mod P and for the implementation of more efficient machine dependent system utility programs for enciphering passwords.

REFERENCES

1. KNOBLE, H.D., FORNEY, C., AND BADER, F.S. An efficient one-way enciphering algorithm. *ACM Trans. Math. Software 5*, 1 (March 1979), 97-107.

ALGORITHM

```
C------------ SAMPLE PROGRAM TO ENCIPHER 72-BIT PASSWORDS ACROSS       000000010
C   MACHINES WITH 3 DIFFERENT WORD SIZES. OUTPUT IS TABLE 2A OR 2B.     000000020
C   TO RUN THIS PROGRAM COMMENTS BEGINNING C32, C36, OR C60 MUST BE     000000030
C   ACTIVATED BY REPLACING THESE FIRST THREE CHARACTERS WITH BLANKS.     000000040
C   TABLE 2A WILL BE PRODUCED ON IBM 360/370 OR XEROX SIGMA SERIES       000000050
C   32-BIT MACHINES BY ACTIVATING COMMENTS BEGINNING WITH C32.           000000060
C   TABLE 2B WILL BE PRODUCED ON UNIVAC 1100, HONEYWELL 600/6000, AND   000000070
C   DEC SYSTEM 10/20 36-BIT MACHINES WHEN ACTIVATING COMMENTS BEGINNING 000000080
C   WITH C36.  TABLE 2B WILL BE PRODUCED ON THE CDC 6600/7600 60-BIT     000000090
C   MACHINES WHEN ACTIVATING COMMENTS BEGINNING WITH C60.                000000100
C                                                                        000000110
C   INPUT TO THIS PROGRAM ARE THE FOLLOWING 10 CARDS BEGINNING WITH C=   000000120
C= HEX(00000000000000000000) OCTAL(00000000000000000000000000000) DECIMAL(0)    000000130
C= HEX(00000000000000000001) OCTAL(00000000000000000000000000001) DECIMAL(1)    000000140
C= HEX(00000000000000000002) OCTAL(00000000000000000000000000002) DECIMAL(2)    000000150
C= HEX(00000000000000000003) OCTAL(00000000000000000000000000003) DECIMAL(3)    000000160
C= HEX(FFFFFFFFFFFFFFFFFA3) OCTAL(77777777777777777777643) DECIMAL(P)    000000170
C= HEX(FFFFFFFFFFFFFFFFFA4) OCTAL(77777777777777777777644) DECIMAL(P+1)  000000180
C= HEX(FFFFFFFFFFFFFFFFFA5) OCTAL(77777777777777777777645) DECIMAL(P+2)  000000190
C= HEX(0000000000000000005C) OCTAL(00000000000000000000134) DECIMAL(92)  000000200
C= HEX(FFFFFFFFFFFFFFFFFFF) OCTAL(77777777777777777777777) DECIMAL(P+92)000000210
C= HEX(5555555555555555) OCTAL(2525252525252525252525) BINARY(0I...)     000000220
```

Received January 1976; revised September 1977.

Author's address: Computation Center, The Pennsylvania State University, Computer Building, University Park, PA 16802.

```
C                                                                       ØØØØØ23Ø
C-----------------------------------------------------------------------ØØØØØ24Ø
      INTEGER A,NN,N(6)                                                 ØØØØØ25Ø
C32   INTEGER X(3),FX(3),P(3),AC(3,6),WORK(43),NP,IW                    ØØØØØ26Ø
C36   INTEGER X(2),FX(2),P(2),AC(2,6),WORK(31),NP,IW                    ØØØØØ27Ø
C6Ø   INTEGER X(2),FX(2),P(2),AC(2,6),WORK(31),NP,IW                    ØØØØØ28Ø
C----------A IS CHOSEN TO MAKE P=2**72-A A PRIME INTEGER IN THE         ØØØØØ29Ø
C           INTERVAL (2**(NP*M)-2**(M-1), 2**(NP*M)). NP*M=72 HERE.     ØØØØØ3ØØ
      DATA A/93/, NN/6/                                                 ØØØØØ31Ø
      DATA N(1)/1Ø48569/,N(2)/524285/,N(3)/3/,N(4)/2/,N(5)/1/,N(6)/Ø/   ØØØØØ32Ø
C.32.....IBM, XEROX 32-BIT MACHINES, HEXADECIMAL CONSTANTS(Z).          ØØØØØ33Ø
C32   DATA NP/3/, IW/43/                                                ØØØØØ34Ø
C32   DATA AC(1,1),AC(2,1),AC(3,1)/Z00000000,Z00000000,Z00000001/,     ØØØØØ35Ø
C32  *     AC(1,2),AC(2,2),AC(3,2)/Z00FFFFFF,Z00FFFFFF,Z00FFFFA1/,     ØØØØØ36Ø
C32  *     AC(1,3),AC(2,3),AC(3,3)/Z00FFFFFF,Z00FFFFFF,Z00FFFF83/,     ØØØØØ37Ø
C32  *     AC(1,4),AC(2,4),AC(3,4)/Z00FFFFFF,Z00FFFFFF,Z00FFFF7B/,     ØØØØØ38Ø
C32  *     AC(1,5),AC(2,5),AC(3,5)/Z00FFFFFF,Z00FFFFFF,Z00FFFF9E/,     ØØØØØ39Ø
C32  *     AC(1,6),AC(2,6),AC(3,6)/Z00FFFFFF,Z00FFFFFF,Z00FFFF02/      ØØØØØ4ØØ
C                                                                       ØØØØØ41Ø
C.36.....UNIVAC, HONEYWELL, DEC 36-BIT MACHINES, OCTAL CONSTANTS(O).    ØØØØØ42Ø
C36   DATA NP/2/, IW/31/                                                ØØØØØ43Ø
C36   DATA AC(1,1),AC(2,1)/Ø00000000000Ø,Ø00000000001/,                 ØØØØØ44Ø
C36  *     AC(1,2),AC(2,2)/Ø77777777777,Ø77777777641/,                 ØØØØØ45Ø
C36  *     AC(1,3),AC(2,3)/Ø77777777777,Ø77777777603/,                 ØØØØØ46Ø
C36  *     AC(1,4),AC(2,4)/Ø77777777777,Ø77777777573/,                 ØØØØØ47Ø
C36  *     AC(1,5),AC(2,5)/Ø77777777777,Ø77777777636/,                 ØØØØØ48Ø
C36  *     AC(1,6),AC(2,6)/Ø77777777777,Ø77777777402/                  ØØØØØ49Ø
C                                                                       ØØØØØ5ØØ
C.6Ø.....CDC 66ØØ/76ØØ 6Ø-BIT +-, 48-BIT */, OCTAL CONSTANTS(B).       ØØØØØ51Ø
C6Ø    DATA NP/2/, IW/31/                                               ØØØØØ52Ø
C6Ø    DATA AC(1,1),AC(2,1)/ØØØØØØØØØØØØØØØØØØØØØB,ØØØØØØØØØØØØØØØØØØØ1B/,ØØØØØ53Ø
C6Ø  *     AC(1,2),AC(2,2)/ØØØØØØØØ777777777777B,ØØØØØØØØ777777777641B/,ØØØØØ54Ø
C6Ø  *     AC(1,3),AC(2,3)/ØØØØØØØØ777777777777B,ØØØØØØØØ777777777603B/,ØØØØØ55Ø
C6Ø  *     AC(1,4),AC(2,4)/ØØØØØØØØ777777777777B,ØØØØØØØØ777777777573B/,ØØØØØ56Ø
C6Ø  *     AC(1,5),AC(2,5)/ØØØØØØØØ777777777777B,ØØØØØØØØ777777777636B/,ØØØØØ57Ø
C6Ø  *     AC(1,6),AC(2,6)/ØØØØØØØØ777777777777B,ØØØØØØØØ777777777402B/  ØØØØØ58Ø
C                                                                       ØØØØØ59Ø
      WRITE(6,1)                                                        ØØØØØ6ØØ
C32 1 FORMAT(45H1HEXADECIMAL INPUT(X)  HEXADECIMAL OUTPUT(FX)/1X)       ØØØØØ61Ø
C36 1 FORMAT(15H1OCTAL INPUT(X),14X,16HOCTAL OUTPUT(FX)/1X)             ØØØØØ62Ø
C6Ø 1 FORMAT(15H1OCTAL INPUT(X),14X,16HOCTAL OUTPUT(FX)/1X)             ØØØØØ63Ø
C--------INPUT AND ENCIPHER 1Ø BIT STRINGS.                            ØØØØØ64Ø
      DO 4 I=1,1Ø                                                       ØØØØØ65Ø
         READ(5,2) X                                                    ØØØØØ66Ø
C32 2    FORMAT(7X,3Z6)                                                 ØØØØØ67Ø
C36 2    FORMAT(33X,2Ø12)                                               ØØØØØ68Ø
C6Ø 2    FORMAT(33X,2Ø12)                                               ØØØØØ69Ø
         CALL PURDY(X,FX,A,P,N,AC,WORK,NP,NN,IW,NP,IERR)                ØØØØØ7ØØ
         IF(IERR.NE.Ø) GO TO 5                                          ØØØØØ71Ø
         WRITE(6,3) X,FX                                                ØØØØØ72Ø
C32 3    FORMAT(1X,3Z6,4X,3Z6)                                          ØØØØØ73Ø
C36 3    FORMAT(1X,2Ø12,4X,2Ø12)                                        ØØØØØ74Ø
C6Ø 3    FORMAT(1X,2Ø12,4X,2Ø12)                                        ØØØØØ75Ø
    4 CONTINUE                                                          ØØØØØ76Ø
      STOP                                                              ØØØØØ77Ø
    5 WRITE(6,6) IERR                                                   ØØØØØ78Ø
    6 FORMAT(46HØ****ERROR RETURN FROM SUBROUTINE PURDY. IERR=,I2)      ØØØØØ79Ø
      STOP                                                              ØØØØØ8ØØ
      END                                                               ØØØØØ81Ø


      BLOCK DATA                                                        ØØØØØ82Ø
      COMMON/MACHIN/ ISIZE,M,MAXPOS,MINNEG,MASK                         ØØØØØ83Ø
C32   DATA ISIZE/32/,M/24/,MAXPOS/Z007FFFFF/,MINNEG/Z00800000/,        ØØØØØ84Ø
C32  *     MASK/Z01000000/                                            ØØØØØ85Ø
C36   DATA ISIZE/36/,M/36/,MAXPOS/0377777777777/,MINNEG/040000000000Ø/, ØØØØØ86Ø
C36  *     MASK/Ø/                                                    ØØØØØ87Ø
C6Ø   DATA ISIZE/6Ø/,M/36/,MAXPOS/ØØØØØØØØ377777777777B/,            ØØØØØ88Ø
C6Ø  *     MINNEG/ØØØØØØØØ4ØØØØØØØØØØØØB/,MASK/ØØØØØØØ1ØØØØØØØØØØØØB/  ØØØØØ89Ø
      END                                                               ØØØØØ9ØØ


      SUBROUTINE PURDY(X,FX,A,PRIME,N,ACOEFF,WORK,NP,NN,NW,IDIMA,IERR)  ØØØØØØ1Ø
C =======ROUTINE TO EVALUATE PURDY'S IRREVERSIBLE ENCIPHERING FUNCTION...ØØØØØØ2Ø
C            FX=F(X)=P(X) MOD PRIME          WHERE                      ØØØØØØ3Ø
C   P(X)= SUM(ACOEFF(I)*X**N(I)), I=1,2,3,...,NN.                       ØØØØØØ4Ø
```

```
C   THAT IS, THIS ROUTINE EVALUATES SPARSE POLYNOMIALS OF VERY LARGE      ØØØØØØ5Ø
C   DEGREE MODULO A PRIME.  SUCH FAMILIES OF ENCIPHERING FUNCTIONS AND    ØØØØØØ6Ø
C   RELATED PROPERTIES ARE DISCUSSED IN...                                ØØØØØØ7Ø
C   A HIGH SECURITY LOG-IN PROCEDURE, BY GEORGE B. PURDY, AUGUST  1974,   ØØØØØØ8Ø
C   CACM 17(8), PAGES 442-444.                                            ØØØØØØ9Ø
C                                                                         ØØØØØ1ØØ
C                                                                         ØØØØØ11Ø
C   PARAMETERS...                                                         ØØØØØ12Ø
C       THE LABELED COMMON/MACHIN/ DESCRIBES PARAMETERS RELATING TO THE   ØØØØØ13Ø
C       HOST MACHINE'S WORD SIZE AND MUST BE INITIALIZED TO THE FOLLOWING ØØØØØ14Ø
C       VALUES BEFORE SUBROUTINE PURDY IS CALLED...                       ØØØØØ15Ø
C                                                                         ØØØØØ16Ø
C    ISIZE   IS GIVEN AS THE TOTAL NUMBER OF BITS IN AN INTEGER COMPUTER  ØØØØØ17Ø
C            WORD.                                                        ØØØØØ18Ø
C                                                                         ØØØØØ19Ø
C    M       IS GIVEN AS AN EVEN INTEGER WHOSE VALUE IS THE NUMBER OF     ØØØØØ2ØØ
C            SIGNIFICANT (LOW-ORDER) BITS IN EACH COMPUTER WORD TO BE     ØØØØØ21Ø
C            USED TO REPRESENT A MULTIPRECISION INTEGER.  +,-,/,* MUST BE ØØØØØ22Ø
C            VALID OVER THE NORMAL RANGES FOR INTEGER ARITHMETIC.         ØØØØØ23Ø
C            THAT IS, INTEGER ARITHMETIC MUST BE VALID OVER               ØØØØØ24Ø
C            -2**(M-1)-1 .LE. R .LE. 2**(M-1)-1  FOR 1'S COMPLEMENT,      ØØØØØ25Ø
C            AND SIGNED MAGNITUDE MACHINES AND VALID OVER THE RANGE       ØØØØØ26Ø
C            -2**(M-1) .LE. R .LE. 2**(M-1)-1 FOR 2'S COMPLEMENT MACHINES ØØØØØ27Ø
C            WHERE R IS ANY COMPUTED RESULT.  FOR SIGNED MAGNITUDE        ØØØØØ28Ø
C            MACHINES M MUST BE EVEN AND NO GREATER THAN ISIZE-2.         ØØØØØ29Ø
C            OTHERWISE M MUST BE EVEN AND LESS THAN OR EQUAL TO ISIZE.    ØØØØØ3ØØ
C                                                                         ØØØØØ31Ø
C    MAXPOS  IS GIVEN AS THE BIT STRING VALUE OF 2**(M-1)-1, THAT IS THE  ØØØØØ32Ø
C            BIT STRING Ø111...111 .                                      ØØØØØ33Ø
C                                                                         ØØØØØ34Ø
C    MINNEG  IS GIVEN AS THE BIT STRING VALUE OF 2**(M-1), THAT IS THE    ØØØØØ35Ø
C            BIT STRING 1ØØØ...ØØØ.                                       ØØØØØ36Ø
C                                                                         ØØØØØ37Ø
C    MASK    IS GIVEN AS THE BIT STRING VALUE OF 2**M MOD 2**ISIZE. THE   ØØØØØ38Ø
C            HOST COMPUTER MUST SUPPORT INTEGER ADDITION AND SUBTRACTION  ØØØØØ39Ø
C            WITH THE VALUE OF MASK AND ALL INTEGERS IN THE RANGE         ØØØØØ4ØØ
C            DESCRIBED FOR M ABOVE.                                       ØØØØØ41Ø
C                                                                         ØØØØØ42Ø
C DEFINITION:  A NP WORD MULTIPRECISION INTEGER IS AN UNSIGNED (NP*M)-    ØØØØØ43Ø
C              BIT NUMBER STORED IN NP SUCCESSIVE INTEGER COMPUTER        ØØØØØ44Ø
C              WORDS, THE HIGHEST ORDER PART BEGINNING WITH THE FIRST     ØØØØØ45Ø
C              WORD, THE LOWEST ORDER ENDING IN THE LAST.  THAT IS,       ØØØØØ46Ø
C              EACH ELEMENT OF THE ARRAY REPRESENTING A MULTIPRECISION    ØØØØØ47Ø
C              INTEGER IS TREATED AS A BASE 2**M DIGIT WITH ISIZE-M       ØØØØØ48Ø
C              LEADING ZERO BITS.  NORMALLY ISIZE=M OR M IS CHOSEN SUCH   ØØØØØ49Ø
C              THAT ISIZE-M IS AN EXACT MULTIPLE OF THE NUMBER OF BITS    ØØØØØ5ØØ
C              USED TO REPRESENT A CHARACTER.  FOLLOWING PURDY'S          ØØØØØ51Ø
C              NOTATION, IN FURTHER DISCUSSIONS HERE Q, THE PASSWORD      ØØØØØ52Ø
C              LENGTH, WILL BE ASSUMED TO BE Q=NP*M.                      ØØØØØ53Ø
C                                                                         ØØØØØ54Ø
C                                                                         ØØØØØ55Ø
C  X        IS GIVEN AS AN UNSIGNED NP-WORD MULTIPRECISION INTEGER        ØØØØØ56Ø
C           REPRESENTING A STRING OF Q BITS TO BE ENCIPHERED.             ØØØØØ57Ø
C           X NORMALLY REPRESENTS A CHARACTER STRING (PASSWORD).          ØØØØØ58Ø
C                                                                         ØØØØØ59Ø
C  FX       IS RETURNED AS AN UNSIGNED NP-WORD MULTIPRECISION INTEGER     ØØØØØ6ØØ
C           WHOSE Q-BIT VALUE IS THE VALUE OF THE ENCIPHERING             ØØØØØ61Ø
C           FUNCTION, F(X), ABOVE.                                        ØØØØØ62Ø
C                                                                         ØØØØØ63Ø
C  A        IS GIVEN AS A POSITIVE INTEGER .LE. MAXPOS WHOSE VALUE IS THE ØØØØØ64Ø
C           PRIME OFFSET CONSTANT DESCRIBED IN THE ABOVE REFERENCE.  THAT ØØØØØ65Ø
C           IS, A MUST BE CHOSEN TO FORCE THE VARIABLE PRIME TO BE PRIME  ØØØØØ66Ø
C           WHERE PRIME IS DEFINED AS PRIME=2**(M*NP) - A.  SOME VALUES   ØØØØØ67Ø
C           ARE TABULATED IN KNUTH, ART OF COMPUTER PROGRAMMING, VOL. 2,  ØØØØØ68Ø
C           TABLE 4.5.4-1.  NO CHECK IS MADE BY THE PROGRAM TO INSURE     ØØØØØ69Ø
C           THIS PRIMALITY CONDITION AS IT ONLY AFFECTS THE KNOWN         ØØØØØ7ØØ
C           DEGENERACY OF THE ENCIPHERING FUNCTION, F.                    ØØØØØ71Ø
C                                                                         ØØØØØ72Ø
C  PRIME    IS RETURNED AS AN UNSIGNED NP-WORD MULTIPRECISION INTEGER     ØØØØØ73Ø
C           WHOSE Q-BIT VALUE IS THE VALUE OF 2**(NP*M)-A.                ØØØØØ74Ø
C                                                                         ØØØØØ75Ø
C  N        IS GIVEN AS AN INTEGER ARRAY OF LENGTH NN WHOSE ELEMENTS ARE  ØØØØØ76Ø
C           THE EXPONENTS OF X.  THE ELEMENTS OF N MUST BE NON-NEGATIVE.  ØØØØØ77Ø
C           TYPICALLY, AS SUGGESTED BY PURDY, THE FIRST ONE OR TWO        ØØØØØ78Ø
C           ELEMENTS OF N ARE VERY LARGE NON-PRIME INTEGERS, SAY          ØØØØØ79Ø
C           GREATER THAN 1Ø**5.                                           ØØØØØ8ØØ
C                                                                         ØØØØØ81Ø
```

```
C   ACOEFF  IS GIVEN AS AN IDIMA BY NN ARRAY REPRESENTING THE UNSIGNED   ØØØØØ82Ø
C           MULTIPRECISION INTEGER COEFFICIENTS OF THE ENCIPHERING       ØØØØØ83Ø
C           POLYNOMIAL F(X) ABOVE.  THESE COEFFICIENTS SHOULD BE         ØØØØØ84Ø
C           CHOSEN TO HAVE (LARGE) VALUES, LESS THAN 2**Q-A.  THE        ØØØØØ85Ø
C           PRECISION OF THE ELEMENTS OF ACOEFF IS NP AND THEREFORE      ØØØØØ86Ø
C           IDIMA USUALLY WOULD BE CHOSEN SUCH THAT IDIMA=NP.            ØØØØØ87Ø
C                                                                        ØØØØØ88Ø
C   WORK    IS GIVEN AS AN WORKSPACE ARRAY OF NW=12*NP+7 INTEGER WORDS.  ØØØØØ89Ø
C                                                                        ØØØØØ9ØØ
C   NP      IS GIVEN AS AN INTEGER WHOSE VALUE IS THE PRECISION OF       ØØØØØ91Ø
C           MULTIPRECISION INTEGERS.  THAT IS, NP IS THE NUMBER OF       ØØØØØ92Ø
C           INTEGER COMPUTER WORDS NECESSARY TO STORE THE UNSIGNED       ØØØØØ93Ø
C           MULTIPRECISION INTEGER 2**Q-1.   NP MUST BE GREATER THAN 1.  ØØØØØ94Ø
C                                                                        ØØØØØ95Ø
C   NN      IS GIVEN AS AN INTEGER WHOSE VALUE IS THE NUMBER OF TERMS IN ØØØØØ96Ø
C           P(X) ABOVE.  NN MUST BE GREATER THAN Ø.                      ØØØØØ97Ø
C                                                                        ØØØØØ98Ø
C   NW      IS GIVEN AS AN INTEGER WHOSE VALUE IS GREATER THAN OR EQUAL  ØØØØØ99Ø
C           TO 12*NP+7.  NW IS THE DIMENSION OF THE WORKSPACE ARRAY, WORK.ØØØØ1ØØØ
C                                                                        ØØØØ1Ø1Ø
C   IDIMA   IS GIVEN AS AN INTEGER WHOSE VALUE IS EQUAL TO THE           ØØØØ1Ø2Ø
C           FIRST DIMENSION CONSTANT OF THE ARGUMENT IN THE CALLING      ØØØØ1Ø3Ø
C           PROGRAM CORRESPONDING TO THE PARAMETER, ACOEFF.  NORMALLY    ØØØØ1Ø4Ø
C           IDIMA=NP.                                                    ØØØØ1Ø5Ø
C                                                                        ØØØØ1Ø6Ø
C   IERR    IS RETURNED AS AN ERROR INDICATOR AS FOLLOWS...              ØØØØ1Ø7Ø
C   IERR=1 IF NP .LE. 1 OR NN IS NOT POSITIVE, IF NW IS LESS THAN 12*NP+7ØØØØ1Ø8Ø
C           OR IF ANY OF THE NN ELEMENTS OF N ARE NEGATIVE.              ØØØØ1Ø9Ø
C   IERR=2 IF A IS NOT IN ITS RANGE DESCRIBED ABOVE.                     ØØØØ11ØØ
C   IERR=3 IF M IS UNEVEN OR GREATER THAN ISIZE OR IF THE HOST MACHINE   ØØØØ111Ø
C           PERFORMS SIGNED MAGNITUDE ARITHMETIC AND M IS UNEVEN OR      ØØØØ112Ø
C           GREATER THAN ISIZE-2.                                        ØØØØ113Ø
C   IERR=4 IF THE LEADING ISIZE-M BITS OF ANY ELEMENT OF X ARE NOT ZERO. ØØØØ114Ø
C   IERR=Ø OTHERWISE. FX AND PRIME ARE RETURNED UNCHANGED IF IERR        ØØØØ115Ø
C           IS RETURNED NON-ZERO.                                        ØØØØ116Ø
C                                                                        ØØØØ117Ø
C   NOTES: THIS ALGORITHM IS PRESENTED HERE AS A GENERAL EVALUATION      ØØØØ118Ø
C           SCHEME AND THE FORTRAN CODE IS INTENDED TO SERVE AS A MODEL  ØØØØ119Ø
C           FOR CONSTRUCTING A SIMILAR SYSTEM UTILITY FUNCTION.          ØØØØ12ØØ
C           TO PROGRAM AN EFFICIENT RE-ENTERABLE ASSEMBLER VERSION OF    ØØØØ121Ø
C           THIS ALGORITHM TO BE USED AS PART OF A SYSTEM PASSWORD       ØØØØ122Ø
C           AUTHORIZATION SCHEME THE FOLLOWING IS RECOMMENDED...         ØØØØ123Ø
C                                                                        ØØØØ124Ø
C    ----   CHOOSE M=ISIZE (EVEN FOR SIGNED MAGNITUDE MACHINES) AND      ØØØØ125Ø
C           USE EQUIVALENT MACHINE INSTRUCTIONS IN PLACE OF SUBROUTINES  ØØØØ126Ø
C           LADD, LSUB, AND KOMP.  ELIMINATE SUBROUTINES SPLIT, JOIN,    ØØØØ127Ø
C           ISIGNM AND ALL REFERENCES TO COMMON BLOCK /MACHIN/.          ØØØØ128Ø
C           IN SUBROUTINE MPMLT CUT THE PRECISION FROM 2*NP TO NP BY USINGØØØØ129Ø
C           THE EQUIVALENT OF A LOGICAL MULTIPLY MACHINE INSTRUCTION.    ØØØØ13ØØ
C                                                                        ØØØØ131Ø
C    ----   CHOOSE A FAMILY OF ENCIPHERING POLYNOMIALS AND FACTOR IT     ØØØØ132Ø
C           BEFORE EVALUATION.  FOR EXAMPLE, IF N(1)=2*N(2)-1, N(3)=3,   ØØØØ133Ø
C           N(4)=2, N(5)=1, N(6)=Ø, NN=6, THEN P(X) FACTORS TO...        ØØØØ134Ø
C           X**(N(2)-1)*(X*(X**(N(2)-1)+ACOEFF(1,1))) +                  ØØØØ135Ø
C           X*(ACOEFF(1,4)+X*(ACOEFF(1,3)+ACOEFF(1,2)*X)).               ØØØØ136Ø
C           THIS REDUCES THE NUMBER OF CALLS TO EXPP THUS SHARPLY REDUCINGØØØØ137Ø
C           COMPUTING TIME.                                              ØØØØ138Ø
C                                                                        ØØØØ139Ø
C    ----   WHEN PRECISION IS CONSTANT, WORKSPACE AND RELATED PARAMETERS  ØØØØ14ØØ
C           BECOME CONSTANTS AND THE PROGRAM IS CONSIDERABLY SIMPLIFIED. ØØØØ141Ø
C           ONCE THIS IS DONE PASS ALL WORKSPACE VARIABLES IN COMMON AND  ØØØØ142Ø
C           REMOVE ALL DIMENSION PARAMETERS AND WORKSPACE PARAMETERS FROM ØØØØ143Ø
C           THE ARGUMENT/PARAMETER LISTS.  THIS SUBSTANTIALLY REDUCES    ØØØØ144Ø
C           LINKAGE TIME.                                                ØØØØ145Ø
C                                                                        ØØØØ146Ø
C  WRITTEN BY H. D. KNOBLE, PENN STATE UNIVERSITY COMPUTATION CENTER,    ØØØØ147Ø
C  JUNE 1977.                                                            ØØØØ148Ø
C=======================================================================ØØØØ149Ø
      INTEGER X(NP),FX(NP),Q,A,PRIME(NP),N(NN),ACOEFF(IDIMA,NN),WORK(NW)ØØØØ15ØØ
      COMMON/MACHIN/ISIZE,M,MAXPOS,MINNEG,MASK                          ØØØØ151Ø
C-----------CHECK PARAMETERS FOR DOMAIN ERRORS.                        ØØØØ152Ø
      IERR=Ø                                                            ØØØØ153Ø
      IF(NP.LE.1.OR.NN.LE.Ø) IERR=1                                     ØØØØ154Ø
      IF(IDIMA.LT.NP.OR.NW.LT.12*NP+7)IERR=1                            ØØØØ155Ø
      IF(A.LE.Ø.OR.A.GT.MAXPOS) IERR=2                                  ØØØØ156Ø
      DO 1 I=1,NN                                                       ØØØØ157Ø
      IF(N(I).LT.Ø) IERR=1                                              ØØØØ158Ø
```

```
1     CONTINUE                                                          ØØØØ159Ø
C------------CHECK HOST MACHINE ASSUMPTIONS ASSUMING MINNEG IS CORRECT. ØØØØ16ØØ
      IF((MOD(M,2).NE.Ø).OR.(M.GT.ISIZE).OR.                            ØØØØ161Ø
     *   (M.GT.ISIZE-2.AND.MINNEG.EQ.Ø)) IERR=3                         ØØØØ162Ø
C------------CHECK IF THE LEADING (ISIZE-M) BITS OF THE WORDS OF X       ØØØØ163Ø
C            ARE NON-ZERO.                                              ØØØØ164Ø
      IF(ISIZE.EQ.M) GO TO 3                                            ØØØØ165Ø
      DO 2 I=1,NP                                                       ØØØØ166Ø
      IF(X(I).GE.MASK) IERR=4                                           ØØØØ167Ø
2     CONTINUE                                                          ØØØØ168Ø
3     IF(IERR.NE.Ø) RETURN                                              ØØØØ169Ø
C-----------FOR Q=NP*M, COMPUTE P=2**Q-A.                              ØØØØ17ØØ
      NP1=NP+1                                                          ØØØØ171Ø
      NP2=NP*2                                                          ØØØØ172Ø
      DO 4 I=1,NP2                                                      ØØØØ173Ø
4     WORK(I)=Ø                                                         ØØØØ174Ø
      WORK(NP2)=A                                                       ØØØØ175Ø
      CALL MPSUB(WORK,WORK(NP1),PRIME,NP)                               ØØØØ176Ø
C----------INITIALIZE                                                  ØØØØ177Ø
      N2=NP2+1                                                          ØØØØ178Ø
      IWE=1Ø*NP+7                                                       ØØØØ179Ø
      IWM=8*NP+7                                                        ØØØØ18ØØ
      IWA=3*NP+2                                                        ØØØØ181Ø
C----------EVALUATE F(X) ONE TERM AT A TIME.                           ØØØØ182Ø
      DO 5 I=1,NN                                                       ØØØØ183Ø
      CALL EXPP(X,N(I),WORK(NP1),PRIME,WORK(N2),NP,IWE)                 ØØØØ184Ø
      CALL MULTP(ACOEFF(1,I),WORK(NP1),WORK(NP1),PRIME,WORK(N2),NP,IWM) ØØØØ185Ø
5     CALL ADDP(WORK,WORK(NP1),WORK,PRIME,WORK(N2),NP,IWA)              ØØØØ186Ø
C-------------SET RESULT = F(X).  FX AND X MAY BE SAME LOCATIONS.       ØØØØ187Ø
      DO 6 I=1,NP                                                       ØØØØ188Ø
6     FX(I)=WORK(I)                                                     ØØØØ189Ø
      RETURN                                                            ØØØØ19ØØ
      END                                                               ØØØØ191Ø


      SUBROUTINE EXPP(X,K,Y,P,WORK,NP,NW)                               ØØØØ192Ø
C========COMPUTE Y=X**K MOD P FOR P=2**Q-A FOR UNSIGNED                 ØØØØ193Ø
C        MULTIPRECISION INTEGERS,X,Y,P OF PRECISION NP.  WORK IS A      ØØØØ194Ø
C        WORKSPACE ARRAY OF NW=NP*1Ø+7 WORDS.                           ØØØØ195Ø
C        SEE KNUTH ALGORITHM 4.6.3-A.                                   ØØØØ196Ø
C                                                                       ØØØØ197Ø
      INTEGER X(NP),Y(NP),P(NP),WORK(NW)                                ØØØØ198Ø
      NPP1=NP+1                                                         ØØØØ199Ø
      IWM=NP*8+7                                                        ØØØØ2ØØØ
      N=K                                                               ØØØØ2Ø1Ø
C---------Y=1, Z=X.                                                    ØØØØ2Ø2Ø
      DO 1 I=1,NP                                                       ØØØØ2Ø3Ø
      Y(I)=Ø                                                            ØØØØ2Ø4Ø
1     WORK(I)=X(I)                                                      ØØØØ2Ø5Ø
      Y(NP)=1                                                           ØØØØ2Ø6Ø
C---------X**Ø=1 FOR X .GE. Ø .                                         ØØØØ2Ø7Ø
      IF(N.LT.1) RETURN                                                 ØØØØ2Ø8Ø
2     L=N                                                               ØØØØ2Ø9Ø
      N=N/2                                                             ØØØØ21ØØ
      IF(MOD(L,2).EQ.Ø) GO TO 5                                         ØØØØ211Ø
      CALL MULTP(Y,WORK,Y,P,WORK(NPP1),NP,IWM)                          ØØØØ212Ø
      IF(N.EQ.Ø) RETURN                                                 ØØØØ213Ø
5     CALL MULTP(WORK,WORK,WORK,P,WORK(NPP1),NP,IWM)                    ØØØØ214Ø
      GO TO 2                                                           ØØØØ215Ø
      END                                                               ØØØØ216Ø


      SUBROUTINE MULTP(R,S,RS,P,WORK,NP,NW)                             ØØØØ217Ø
C=========COMPUTE RS=R*S MOD P FOR P=2**Q-A FOR UNSIGNED                ØØØØ218Ø
C         MULTIPRECISION INTEGERS R,S,RS,P OF PRECISION NP.  WORK       ØØØØ219Ø
C         IS A WORKSPACE ARRAY OF NW=NP*8+7 WORDS.                      ØØØØ22ØØ
      INTEGER R(NP),S(NP),RS(NP),P(NP),WORK(NW)                         ØØØØ221Ø
      NP2=2*NP                                                          ØØØØ222Ø
      IW1=NP2+1                                                         ØØØØ223Ø
      IW2=IW1+NP2                                                       ØØØØ224Ø
      IW3=IW2+NP2                                                       ØØØØ225Ø
      CALL MPMLT(R,S,WORK,WORK(IW1),WORK(IW2),WORK(IW3),NP,NP,          ØØØØ226Ø
     * NP2,NP2,NP2,2*NP2)                                              ØØØØ227Ø
      CALL MOD2Q(WORK,RS,P,WORK(IW1),NP2,NP,NP*6+7)                     ØØØØ228Ø
      RETURN                                                            ØØØØ229Ø
      END                                                               ØØØØ23ØØ
```

```
      SUBROUTINE ADDP(R,S,RPS,P,WORK,NP,NW)                              ØØØØ231Ø
C=========COMPUTE RPS=R+S MOD P FOR P=2**Q-A FOR UNSIGNED                ØØØØ232Ø
C         MULTIPRECISION INTEGERS R,S,RPS,P OF PRECISION NP.             ØØØØ233Ø
C         WORK IS A WORKSPACE ARRAY OF NW=NP*3+2 WORDS.                  ØØØØ234Ø
      INTEGER R(NP),S(NP),RPS(NP),P(NP),WORK(NW)                         ØØØØ235Ø
      NPP1=NP+1                                                          ØØØØ236Ø
      CALL MPADD(R,S,WORK,NP,NPP1)                                       ØØØØ237Ø
      CALL MODQ1(WORK,RPS,P,WORK(NPP1+1),NPP1,NP,2*NP+1)                 ØØØØ238Ø
      RETURN                                                             ØØØØ239Ø
      END                                                                ØØØØ24ØØ


      SUBROUTINE MOD2Q(W,R,P,WORK,NP2,NP,NW)                             ØØØØ241Ø
C=========COMPUTE R=W MOD P FOR 2Q-BIT W, P=2**Q-A FOR UNSIGNED          ØØØØ242Ø
C         MULTIPRECISION INTEGERS W OF PRECISION NP2=NP*2,               ØØØØ243Ø
C         AND R,P OF PRECISION NP.  WORK IS A WORKSPACE ARRAY OF         ØØØØ244Ø
C         NW=NP*6+7 WORDS.                                               ØØØØ245Ø
      INTEGER W(NP2),R(NP),P(NP),WORK(NW),U,A,JP                         ØØØØ246Ø
      COMMON /MACHIN/ISIZE,M,MAXPOS,MINNEG,MASK                          ØØØØ247Ø
      NPP1=NP+1                                                          ØØØØ248Ø
      NPM2=NPP1*2                                                        ØØØØ249Ø
      U=1                                                                ØØØØ25ØØ
      JP=U+NPP1                                                          ØØØØ251Ø
      A=JP                                                               ØØØØ252Ø
      IWU=JP+NPP1                                                        ØØØØ253Ø
      IW1=IWU+1                                                          ØØØØ254Ø
      IW2=IW1+2                                                          ØØØØ255Ø
      IW3=IW2+NP2                                                        ØØØØ256Ø
C-----------FORM U=A*W1.                                                ØØØØ257Ø
      WORK(A)= MASK-P(NP)                                                ØØØØ258Ø
      CALL MPMLT(WORK(A),W,WORK(U),WORK(IW1),WORK(IW2),WORK(IW3),        ØØØØ259Ø
     * 1,NP,NPP1,2,NP2,NPM2)                                            ØØØØ26ØØ
C---------------USE KNUTH'S THEOREM 4.3.1.B AND LEADING DIGIT THEOREM    ØØØØ261Ø
C               TO SOLVE FOR J SUCH THAT J*P.LE. U .LT.(J+1)*P.          ØØØØ262Ø
      R(1)=WORK(U)+1                                                     ØØØØ263Ø
      CALL MPMLT(R(1),P,WORK(JP),WORK(IW1),WORK(IW2),WORK(IW3),          ØØØØ264Ø
     * 1,NP,NPP1,2,NP2,NPM2)                                            ØØØØ265Ø
      IF(KOMP(WORK(JP),WORK(U),NPP1).LE.Ø) GO TO 1                       ØØØØ266Ø
      CALL MPMLT(WORK(U),P,WORK(JP),WORK(IW1),WORK(IW2),WORK(IW3),       ØØØØ267Ø
     * 1,NP,NPP1,2,NP2,NPM2)                                            ØØØØ268Ø
C----------------COMPUTE U MOD 2**Q - JP MOD 2**Q.                       ØØØØ269Ø
1     CALL MPSUB(WORK(U+1),WORK(JP+1),R,NP)                              ØØØØ27ØØ
      CALL ADDP(W(NPP1),R,R,P,WORK(IW1),NP,NPM2+NP)                      ØØØØ271Ø
      RETURN                                                             ØØØØ272Ø
      END                                                                ØØØØ273Ø


      SUBROUTINE MODQ1(S,Y,P,WORK,NP1,NP,NW)                             ØØØØ274Ø
C===========COMPUTE Y=S MOD P FOR (Q+1)-BIT S AND P=2**Q-A FOR UNSIGNED  ØØØØ275Ø
C           MULTIPRECISION INTEGERS S OF PRECISION NP1 AND               ØØØØ276Ø
C           Y,P OF PRECISION NP.  WORK IS A WORKSPACE OF NW=2*NP+1       ØØØØ277Ø
C           WORDS.                                                       ØØØØ278Ø
      INTEGER S(NP1),Y(NP),P(NP),WORK(NW)                                ØØØØ279Ø
      COMMON/MACHIN/ISIZE,M,MAXPOS,MINNEG,MASK                           ØØØØ28ØØ
      CALL MODQ(S(2),Y,P,NP)                                             ØØØØ281Ø
C-------------S(1), THE HIGH-ORDER PART OF S, IS EITHER ZERO OR ONE.     ØØØØ282Ø
      IF(S(1).EQ.Ø) RETURN                                               ØØØØ283Ø
C-------------RECOVER A=2**Q-P                                           ØØØØ284Ø
      DO 1 I=1,NP                                                        ØØØØ285Ø
1     WORK(I)=Ø                                                          ØØØØ286Ø
      WORK(NP)=MASK-P(NP)                                                ØØØØ287Ø
C-------------COMPUTE SUM=MODQ(S MOD 2**Q)+A*S(1).                       ØØØØ288Ø
      CALL MPADD(Y,WORK,WORK(NP1),NP,NP1)                                ØØØØ289Ø
      CALL MODQ(WORK(NP1+1),Y,P,NP)                                      ØØØØ29ØØ
      RETURN                                                             ØØØØ291Ø
      END                                                                ØØØØ292Ø


      SUBROUTINE MODQ(X,Y,P,NP)                                          ØØØØ293Ø
C===========COMPUTE X MOD P FOR Q-BIT X AND P=2**Q-A FOR UNSIGNED        ØØØØ294Ø
C           MULTIPRECISION INTEGERS X,Y,P OF PRECISION NP.               ØØØØ295Ø
      INTEGER X(NP),Y(NP),P(NP)                                          ØØØØ296Ø
      IF(KOMP(X,P,NP).GE.Ø) GO TO 2                                      ØØØØ297Ø
C------------Y=X.                                                       ØØØØ298Ø
      DO 1 I=1,NP                                                        ØØØØ299Ø
1     Y(I)=X(I)                                                          ØØØØ3ØØØ
```

```
      RETURN                                                            ØØØØ3Ø1Ø
C---------Y=X-P                                                        ØØØØ3Ø2Ø
2     CALL MPSUB(X,P,Y,NP)                                              ØØØØ3Ø3Ø
      RETURN                                                            ØØØØ3Ø4Ø
      END                                                               ØØØØ3Ø5Ø


      SUBROUTINE MPMLT(IX,IY,IZ,U,V,W,NX,MY,NPM,N2,M2,NPM2)             ØØØØ3Ø6Ø
C==========UNSIGNED MULTIPRECISION INTEGER MULTIPLICATION              ØØØØ3Ø7Ø
C           IZ=IX*IY.  NX,MX,NPM ARE THE PRECISIONS OF THE UNSIGNED   ØØØØ3Ø8Ø
C           INTEGERS IX,IY,IZ RESPECTIVELY.  U,V,W ARE                ØØØØ3Ø9Ø
C           WORKSPACES OF N2=NX*2, M2=MY*2, AND MPN2=(NX+MY)*2 WORDS  ØØØØ31ØØ
C           RESPECTIVELY.  NO INTEGER OVERFLOWS ARE GENERATED.        ØØØØ311Ø
C                                                                     ØØØØ312Ø
C           SEE KNUTH'S ALGORITHM 4.3.1 M.                            ØØØØ313Ø
C                                                                     ØØØØ314Ø
      INTEGER IX(NX),IY(MY),IZ(NPM),U(N2),V(M2),W(NPM2)                 ØØØØ315Ø
      INTEGER T(2),K,SHIFT1,VØ                                          ØØØØ316Ø
      COMMON /MACHIN/ ISIZE,M,MAXPOS,MINNEG,MASK                        ØØØØ317Ø
      SHIFT1=2**(M/2-1)                                                 ØØØØ318Ø
C----------DOUBLE THE PRECISION SINCE FORTRAN CANNOT ACCESS THE         ØØØØ319Ø
C          PRODUCT OF TWO M-BIT WORDS.                                  ØØØØ32ØØ
      DO 1Ø I=1,NX                                                      ØØØØ321Ø
      I2=I*2                                                            ØØØØ322Ø
1Ø    CALL SPLIT(IX(I),U(I2-1),U(I2))                                   ØØØØ323Ø
      DO 11 I=1,MY                                                      ØØØØ324Ø
      I2=I*2                                                            ØØØØ325Ø
11    CALL SPLIT(IY(I),V(I2-1),V(I2))                                   ØØØØ326Ø
C----------BEGIN KNUTH'S ALGORITHM.                                     ØØØØ327Ø
      DO 1 I=1,NPM2                                                     ØØØØ328Ø
1     W(I)=Ø                                                            ØØØØ329Ø
      J=M2                                                              ØØØØ33ØØ
C----------THE PROBABILITY OF V(J)=Ø IS SMALL. SKIP STEP 2.             ØØØØ331Ø
3     I=N2                                                              ØØØØ332Ø
      K=Ø                                                               ØØØØ333Ø
4     IPJ=I+J                                                           ØØØØ334Ø
C----------COMPUTE T=U(I)*V(J)+W(I+J)+K WITHOUT INTEGER OVERFLOW.       ØØØØ335Ø
      VØ=MOD(V(J),SHIFT1)                                               ØØØØ336Ø
      CALL LADD(U(I)*VØ, (V(J)-VØ)*U(I), IZ)                            ØØØØ337Ø
      CALL LADD(IZ(2), W(IPJ)+K, T)                                     ØØØØ338Ø
C-----------COMPUTE K = FLOOR(T/2**(M/2)), W(IPJ)=T MOD 2**(M/2),       ØØØØ339Ø
C                  THE HIGH AND LOW-ORDER HALFS OF T(2).                ØØØØ34ØØ
      CALL SPLIT(T(2),K,W(IPJ))                                         ØØØØ341Ø
5     I=I-1                                                             ØØØØ342Ø
      IF(I.GT.Ø) GO TO 4                                                ØØØØ343Ø
      W(J)=K                                                            ØØØØ344Ø
6     J=J-1                                                             ØØØØ345Ø
      IF(J.GT.Ø) GO TO 3                                                ØØØØ346Ø
C----------CONVERT THE RADIX 2**(M/2) DIGITS TO RADIX 2**M DIGITS.      ØØØØ347Ø
      DO 12 I=1,NPM                                                     ØØØØ348Ø
      I2=I*2                                                            ØØØØ349Ø
12    IZ(I)=JOIN(W(I2-1),W(I2))                                         ØØØØ35ØØ
      RETURN                                                            ØØØØ351Ø
      END                                                               ØØØØ352Ø


      SUBROUTINE MPADD(U,V,W,N,NP1)                                     ØØØØ353Ø
C============UNSIGNED MULTIPRECISION INTEGER ADDITION,                 ØØØØ354Ø
C             W=U+V.  N IS THE PRECISION OF UNSIGNED NUMBERS, IX, IY.   ØØØØ355Ø
C             W IS THE UNSIGNED NP1-WORD PRODUCT.                       ØØØØ356Ø
C                                                                       ØØØØ357Ø
C             SEE KNUTH'S ALGORITHM 4.3.1-A.                            ØØØØ358Ø
      INTEGER U(N),V(N),W(NP1),SUM1(2),SUM2(2)                          ØØØØ359Ø
1     J=N                                                               ØØØØ36ØØ
      K=Ø                                                               ØØØØ361Ø
2     CALL LADD(U(J),V(J),SUM1)                                         ØØØØ362Ø
      CALL LADD(SUM1(2),K,SUM2)                                         ØØØØ363Ø
      K=SUM1(1)+SUM2(1)                                                 ØØØØ364Ø
      W(J+1)=SUM2(2)                                                    ØØØØ365Ø
3     J=J-1                                                             ØØØØ366Ø
      IF(J.GT.Ø) GO TO 2                                                ØØØØ367Ø
      W(1)=K                                                            ØØØØ368Ø
      RETURN                                                            ØØØØ369Ø
      END                                                               ØØØØ37ØØ
```

```
      SUBROUTINE MPSUB(U,V,W,N)                                          ØØØØ371Ø
C===========UNSIGNED MULTIPRECISION INTEGER SUBTRACTION,                  ØØØØ372Ø
C           W=U-V IGNORING POSSIBLE BORROW IF U IS LESS THAN V.           ØØØØ373Ø
C           N IS THE PRECISION OF UNSIGNED NUMBERS U,V,W.                 ØØØØ374Ø
C                                                                         ØØØØ375Ø
C           SEE KNUTH'S ALGORITHM 4.3.1-S.                                ØØØØ376Ø
      INTEGER U(N),V(N),W(N),DIF1(2),DIF2(2)                              ØØØØ377Ø
1     J=N                                                                 ØØØØ378Ø
      K=Ø                                                                 ØØØØ379Ø
2     CALL LSUB(U(J),V(J),DIF1)                                           ØØØØ38ØØ
      CALL LSUB(DIF1(2),K,DIF2)                                           ØØØØ381Ø
      K=DIF1(1)+DIF2(1)                                                   ØØØØ382Ø
      W(J)=DIF2(2)                                                        ØØØØ383Ø
3     J=J-1                                                               ØØØØ384Ø
      IF(J.GT.Ø) GO TO 2                                                  ØØØØ385Ø
      RETURN                                                              ØØØØ386Ø
      END                                                                 ØØØØ387Ø


      FUNCTION KOMP(U,V,N)                                                ØØØØ388Ø
CXXXXXXXXX FUNCTION TO COMPARE UNSIGNED MULTIPRECISION                    ØØØØ389Ø
C           INTEGERS U,V OF PRECISION N.  KOMP=-1 IF U IS LESS THAN V,    ØØØØ39ØØ
C          +1 IF U IS GREATER THAN V, AND Ø IF U=V.                       ØØØØ391Ø
C                                                                         ØØØØ392Ø
C           SEE KNUTH PROBLEM 4.3.1-11                                    ØØØØ393Ø
      INTEGER U(N),V(N)                                                   ØØØØ394Ø
      DO 1 J=1,N                                                          ØØØØ395Ø
C----------IF HIGH-ORDER BITS DIFFER U .NE. V.                           ØØØØ396Ø
      IF(ISIGN(1,U(J)).NE.ISIGN(1,V(J))) GO TO 2                          ØØØØ397Ø
C----------ELSE IF U IS LESS THAN V, THEN KOMP=-1                        ØØØØ398Ø
      IF(U(J).LT.V(J)) GO TO 3                                            ØØØØ399Ø
C----------ELSE IF U IS GREATER THAN V THEN KOMP=+1.                      ØØØØ4ØØØ
      IF(U(J).GT.V(J)) GO TO 4                                            ØØØØ4Ø1Ø
1     CONTINUE                                                            ØØØØ4Ø2Ø
C----------U=V.                                                          ØØØØ4Ø3Ø
      KOMP=Ø                                                              ØØØØ4Ø4Ø
      RETURN                                                              ØØØØ4Ø5Ø
C----------THE OPERAND WITH HIGH-ORDER BIT ON IS GREATER.                 ØØØØ4Ø6Ø
2     IF(U(J).LT.V(J))GO TO 4                                             ØØØØ4Ø7Ø
3     KOMP=-1                                                             ØØØØ4Ø8Ø
      RETURN                                                              ØØØØ4Ø9Ø
4     KOMP=+1                                                             ØØØØ41ØØ
      RETURN                                                              ØØØØ411Ø
      END                                                                 ØØØØ412Ø


      FUNCTION JOIN(HALF1,HALF2)                                          ØØØØ413Ø
CXXXXXXXXXX ROUTINE TO CONVERT TWO RADIX 2**(M/2) DIGITS HALF1,           ØØØØ414Ø
C           HALF2 TO A RADIX 2**M DIGIT, JOIN.  THAT IS THIS              ØØØØ415Ø
C           ROUTINE FORMS A M-BIT NUMBER WITH THE HIGH-ORDER ISIZE-M      ØØØØ416Ø
C           BITS EQUAL TO ZERO, THE M/2 HIGH-ORDER BITS BEING THE M/2     ØØØØ417Ø
C           LOW-ORDER BITS OF HALF1, AND THE M/2 LOW-ORDER BITS BEING     ØØØØ418Ø
C           THE M/2 LOW-ORDER BITS OF HALF2.                              ØØØØ419Ø
C           THIS IS DONE WITHOUT CAUSING AN INTEGER OVERFLOW AND          ØØØØ42ØØ
C           ASSUMING M IS EVEN, AND HIGH-ORDER (ISIZE-M)+M/2 BITS OF      ØØØØ421Ø
C           EACH HALF ARE ZERO.                                           ØØØØ422Ø
      INTEGER HALF1,HALF2,ISUM(2),SHIFT1                                  ØØØØ423Ø
      COMMON/MACHIN/ISIZE,M,MAXPOS,MINNEG,MASK                            ØØØØ424Ø
      SHIFT1=2**((M/2)-1)                                                 ØØØØ425Ø
C-----------FORM JOIN=HALF2+(2**(M/2))*HALF1                             ØØØØ426Ø
      IX=HALF1*SHIFT1                                                     ØØØØ427Ø
      CALL LADD(IX,IX,ISUM)                                               ØØØØ428Ø
      JOIN=ISUM(2)+HALF2                                                  ØØØØ429Ø
      RETURN                                                              ØØØØ43ØØ
      END                                                                 ØØØØ431Ø


      SUBROUTINE SPLIT(IX,HALF1,HALF2)                                    ØØØØ432Ø
CXXXXXXXXXXX ROUTINE TO CONVERT A RADIX 2**M DIGIT, IX TO TWO             ØØØØ433Ø
C             RADIX 2**(M/2) DIGITS HALF1,HALF2.  THAT IS THIS            ØØØØ434Ø
C             ROUTINE SPLITS THE M-BIT NUMBER IX INTO HIGH-ORDER HALF     ØØØØ435Ø
C             HALF1 AND LOW-ORDER HALF2 WITHOUT CAUSING AN INTEGER        ØØØØ436Ø
C             OVERFLOW. THE HIGH-ORDER ISIZE-M BITS OF IX ARE ASSUMED     ØØØØ437Ø
C             TO BE ZERO.                                                 ØØØØ438Ø
      INTEGER X,HALF1,HALF2,SHIFT,ISUM(2)                                 ØØØØ439Ø
      COMMON/MACHIN/ISIZE,M,MAXPOS,MINNEG,MASK                            ØØØØ44ØØ
```

```
      SHIFT=2**(M/2)                                                    ØØØØ441Ø
      X=IX                                                              ØØØØ442Ø
      IF(X.GE.Ø.AND.X.LE.MAXPOS) GO TO 2                                ØØØØ443Ø
C-----------REMOVE SIGN BIT BEFORE DIVISION.                           ØØØØ444Ø
      CALL LADD(MINNEG,X,ISUM)                                          ØØØØ445Ø
      HALF1=ISUM(2)/SHIFT+SHIFT/2                                       ØØØØ446Ø
C-----------FORM X MOD 2**(M/2)                                        ØØØØ447Ø
      HALF2=MOD(ISUM(2),SHIFT)                                          ØØØØ448Ø
      RETURN                                                            ØØØØ449Ø
2     HALF1=X/SHIFT                                                     ØØØØ45ØØ
      HALF2=MOD(X,SHIFT)                                                ØØØØ451Ø
      RETURN                                                            ØØØØ452Ø
      END                                                               ØØØØ453Ø


      SUBROUTINE LSUB(IU,IV,IW)                                         ØØØØ454Ø
CXXXXXXXXX SUBROUTINE TO PERFORM A LOGICAL SUBTRACT OF 2 M-BIT          ØØØØ455Ø
C          NUMBERS IU, IV.  IW(2) IS THEIR LOGICAL DIFFERENCE,          ØØØØ456Ø
C          IW(1) IS +1 IF THERE WAS A BORROW, ZERO OTHERWISE.           ØØØØ457Ø
C          SEE SUBROUTINE LADD.                                         ØØØØ458Ø
      INTEGER IU,IV,IX,IY,IW(2),ISUM(2)                                 ØØØØ459Ø
      COMMON /MACHIN/ISIZE,M,MAXPOS,MINNEG,MASK                         ØØØØ46ØØ
      IX=IU                                                             ØØØØ461Ø
      IY=IV                                                             ØØØØ462Ø
      IW(1)=Ø                                                           ØØØØ463Ø
C----------SORT OUT SPECIAL CASES FOR LADD.                            ØØØØ464Ø
      IF(IX.EQ.IY) GO TO 3                                              ØØØØ465Ø
      IF(IY.EQ.Ø) GO TO 1                                               ØØØØ466Ø
      IF(IY.EQ.MINNEG) GO TO 2                                          ØØØØ467Ø
      CALL LADD(IX,MASK-IY,ISUM)                                        ØØØØ468Ø
      IF(ISUM(1).EQ.Ø) IW(1)=1                                          ØØØØ469Ø
      IW(2)=ISUM(2)                                                     ØØØØ47ØØ
      RETURN                                                            ØØØØ471Ø
1     IW(2)=IX                                                          ØØØØ472Ø
      RETURN                                                            ØØØØ473Ø
2     CALL LADD(IX,IY,ISUM)                                             ØØØØ474Ø
      IF(ISUM(1).EQ.Ø) IW(1)=1                                          ØØØØ475Ø
      IW(2)=ISUM(2)                                                     ØØØØ476Ø
      RETURN                                                            ØØØØ477Ø
3     IW(2)=Ø                                                           ØØØØ478Ø
      RETURN                                                            ØØØØ479Ø
      END                                                               ØØØØ48ØØ


      SUBROUTINE LADD(IU,IV,IW)                                         ØØØØ481Ø
CXXXXXXXXXXXXX SUBROUTINE TO PERFORM A LOGICAL ADD OF THE LOW-ORDER     ØØØØ482Ø
C              M BITS OF TWO COMPUTER WORDS, IU, IV.  IW(2) IS THEIR    ØØØØ483Ø
C              LOGICAL SUM, IW(1) IS THE CARRY BIT VALUE.               ØØØØ484Ø
C              NO INTEGER OVERFLOWS OCCUR EVEN IF M=ISIZE.              ØØØØ485Ø
C              EXAMPLE -  Ø111 LADD 1Ø11 YIELDS IW(1)=1, IW(2)=ØØ1Ø,    ØØØØ486Ø
C              FOR M=4, MAXPOS=Ø111, MINNEG=1ØØØ, MASK=Ø OR 1ØØØØ.      ØØØØ487Ø
C                                                                       ØØØØ488Ø
      INTEGER IX,IY,U,IV,IW(2),DIF,NEG,POS,ISUM,BIG,SMALL,ISX,ISY,ISD   ØØØØ489Ø
      COMMON/MACHIN/ISIZE,M,MAXPOS,MINNEG,MASK                          ØØØØ49ØØ
      IX=IU                                                             ØØØØ491Ø
      IY=IV                                                             ØØØØ492Ø
      IW(1)=Ø                                                           ØØØØ493Ø
C-------------ELIMINATE ZERO OPERANDS.                                 ØØØØ494Ø
      IF(IX.EQ.Ø.OR.IY.EQ.Ø) GO TO 2                                    ØØØØ495Ø
C-------------DETERMINE IF IX,IY IS ++, --, OR +-                      ØØØØ496Ø
      ISX=ISIGNM(IX)                                                    ØØØØ497Ø
      ISY=ISIGNM(IY)                                                    ØØØØ498Ø
      IF(ISX.NE.ISY) GO TO 5                                            ØØØØ499Ø
      IF(ISX.LT.Ø) GO TO 3                                              ØØØØ5ØØØ
C-------------++ (BOTH OPERANDS HAVE HIGH-ORDER BITS OFF)               ØØØØ5Ø1Ø
1     DIF=MINØ(IX,IY)-(MAXPOS-MAXØ(IX,IY))                             ØØØØ5Ø2Ø
      ISD=ISIGNM(DIF)                                                   ØØØØ5Ø3Ø
      IF(ISD.LE.Ø) IW(2)=IX+IY                                          ØØØØ5Ø4Ø
      IF(ISD.GT.Ø) IW(2)=(MINNEG+DIF)-1                                 ØØØØ5Ø5Ø
      RETURN                                                            ØØØØ5Ø6Ø
C------------AT LEAST ONE OPERAND IS ZERO.                            ØØØØ5Ø7Ø
2     IW(2)=IX+IY                                                       ØØØØ5Ø8Ø
      RETURN                                                            ØØØØ5Ø9Ø
C------------ -- (BOTH OPERANDS HAVE HIGH-ORDER BITS ON)               ØØØØ51ØØ
3     IW(1)=1                                                           ØØØØ511Ø
      BIG=MAXØ(IX,IY)                                                   ØØØØ512Ø
```

```
      SMALL=MINØ(IX,IY)                                                  ØØØØ513Ø
      DIF=Ø                                                              ØØØØ514Ø
      IF(SMALL.NE.MINNEG) DIF=MINNEG-SMALL                               ØØØØ515Ø
      DIF=BIG-DIF                                                        ØØØØ516Ø
      IF(ISIGNM(DIF).GE.Ø) GO TO 4                                       ØØØØ517Ø
      IW(2)=((MAXPOS+DIF)+1)-MASK                                        ØØØØ518Ø
      RETURN                                                             ØØØØ519Ø
4     IW(2)=IX+IY                                                        ØØØØ52ØØ
      RETURN                                                             ØØØØ521Ø
C------------ +- (OPERANDS HAVE OPPOSITE HIGH-ORDER BIT VALUES).         ØØØØ522Ø
5     IW(2)=IX+IY                                                        ØØØØ523Ø
      NEG=IX                                                             ØØØØ524Ø
      POS=IY                                                             ØØØØ525Ø
      IF(ISIGNM(IX).LT.Ø) GO TO 6                                        ØØØØ526Ø
      NEG=IY                                                             ØØØØ527Ø
      POS=IX                                                             ØØØØ528Ø
C------------ IF ++(--(MINNEG,NEG),POS) IMPLIES A ++ INTEGER OVERFLOW    ØØØØ529Ø
C             THEN THERE WILL BE A CARRY FOR +-.                         ØØØØ53ØØ
6     ISUM=((MAXPOS+NEG)+1)-MASK                                         ØØØØ531Ø
      IF(MINØ(ISUM,POS).LE.(MAXPOS-MAXØ(ISUM,POS))) RETURN              ØØØØ532Ø
      IW(2)=IW(2)-MASK                                                   ØØØØ533Ø
      IW(1)=1                                                            ØØØØ534Ø
      RETURN                                                             ØØØØ535Ø
      END                                                                ØØØØ536Ø


      FUNCTION ISIGNM(IX)                                                ØØØØ537Ø
CXXXXXXXX FUNCTION TO RETURN Ø IF IX=Ø, 1 IF ØFIXFMAXPOS+1,-1 OTHERWISE.ØØØØ538Ø
      COMMON /MACHIN/ISIZE,M,MAXPOS,MINNEG,MASK                          ØØØØ539Ø
      ISIGNM=1                                                           ØØØØ54ØØ
      IF(IX.LT.Ø.OR.IX.GT.MAXPOS)ISIGNM=-1                               ØØØØ541Ø
      IF(IX.EQ.Ø) ISIGNM=Ø                                               ØØØØ542Ø
      RETURN                                                             ØØØØ543Ø
      END                                                                ØØØØ544Ø
```

537-P 1- 0

ALGORITHM 537
Characteristic Values of Mathieu's Differential Equation [S22]

WALTER R. LEEB
Technische Universität Wien, Austria

Key Words and Phrases: Mathieu's differential equation, wave equation, characteristic values, eigenvalues, separation constants, Mathieu functions, ordinary Mathieu functions, modified Mathieu functions, elliptic cylinder functions, hyperbolic cylinder functions
CR Categories: 5.14, 5.17
Language: Fortran

DESCRIPTION

The algorithm calculates the characteristic values b of Mathieu's differential equation

$$d^2y/dx^2 + (b - S\cos^2 x)\,y = 0 \qquad (1)$$

for solutions $y(x)$ of periodicity π or 2π. In this case b is a function of S and belongs to a countably infinite set of values for every S [2]. The knowledge of b is a prerequisite for calculating ordinary and modified Mathieu functions.

The program has been tested for real S-parameters between zero and 1000, and for a number of characteristic values of up to 24, depending on the value of S. (For negative S, simple equations exist which relate the corresponding b to those for positive S [2, p. xvii]). The calculated b are correct to at least 9 decimal places. Within the calculated finite subset of b, the characteristic values are arranged with increasing magnitude, the first value being the smallest of the infinite set. The algorithm uses a library subroutine to find the eigenvalues of a real symmetric tridiagonal matrix.

Four types of solutions $y(x)$ can be distinguished: An odd and an even solution with period 2π, and an odd and an even solution with period π. The corresponding characteristic values b are called bo_{2r+1}, be_{2r+1}, bo_{2r+2},[1] and be_{2r}. ($r = 0, 1, 2, \ldots$; within each of the four sets, the smallest index is used for the smallest value of b.) Upon insertion of $y(x)$ in the form of Fourier series into eq. (1), a system of homogeneous, linear equations for the Fourier coefficients is obtained [2]. The determinant of the resulting square coefficient matrix of infinite order has to equal zero for nontrivial $y(x)$. This characteristic equation determining the

[1] Note the difference in the index notation for bo_{2r+2} compared with [2], where this set of characteristic values is called bo_{2r}. We have thus avoided unequal starting values for r among the four sets.

Received 27 December 1976, 6 January 1978, and 22 May 1978.

This research was supported by the Fonds zur Förderung der wissenschaftlichen Forschung, Austria.
Author's address: Institut für Hochfrequenztechnik, Technische Universität Wien, Gusshausstrasse 25, A-1040 Wien, Austria.

characteristic values b may be written as

$$|\beta I - A| = 0, \tag{2}$$

where $\beta = b - S/2$ and I is the identity matrix. In the case of y being odd with a periodicity of 2π, the elements of the matrix A are found to be

$$a_{11} = 1 - S/4;\ a_{i-1,i} = a_{i,i-1} = S/4,\ a_{ii} = (2i-1)^2 \quad \text{for} \quad i \geq 2,$$

and all the other a_{ij} are equal to zero. Therefore, after adding $S/2$, the eigenvalues of the real symmetric tridiagonal matrix A (of unrestricted order) yield the characteristic values bo_{2r+1}.

Correspondingly, one can show that the be_{2r+1}, bo_{2r+2}, and be_{2r} are given by the eigenvalues of matrices which we call B, C and D, respectively. The nonzero elements of all four matrices have been compiled in Table I. When proceeding the way outlined above, the matrix D' determining the characteristic values be_{2r}

Table I. Nonzero Elements for Calculation of Characteristic Values

Characteristic value	Matrix elements
bo_{2r+1}	$a_{11} = 1 - S/4$; $a_{i-1,i} = a_{i,i-1} = S/4$ and $a_{ii} = (2i-1)^2$ for $i \geq 2$
be_{2r+1}	$b_{11} = 1 + S/4$; $b_{i-1,i} = b_{i,i-1} = S/4$ and $b_{ii} = (2i-1)^2$ and $i \geq 2$
bo_{2r+2}	$c_{ii} = 4t^2$ for $i \geq 1$; $c_{i,i-1} = c_{i-1,i} = S/4$ for $i \geq 2$
be_{2r}	$d_{12} = d_{21} = S/(8^{1/2})$; $d_{22} = 4$
	$d_{ii} = 4(i-1)^2$ and $d_{i-1,i} = d_{i,i-1} = S/4$ for $i \geq 3$

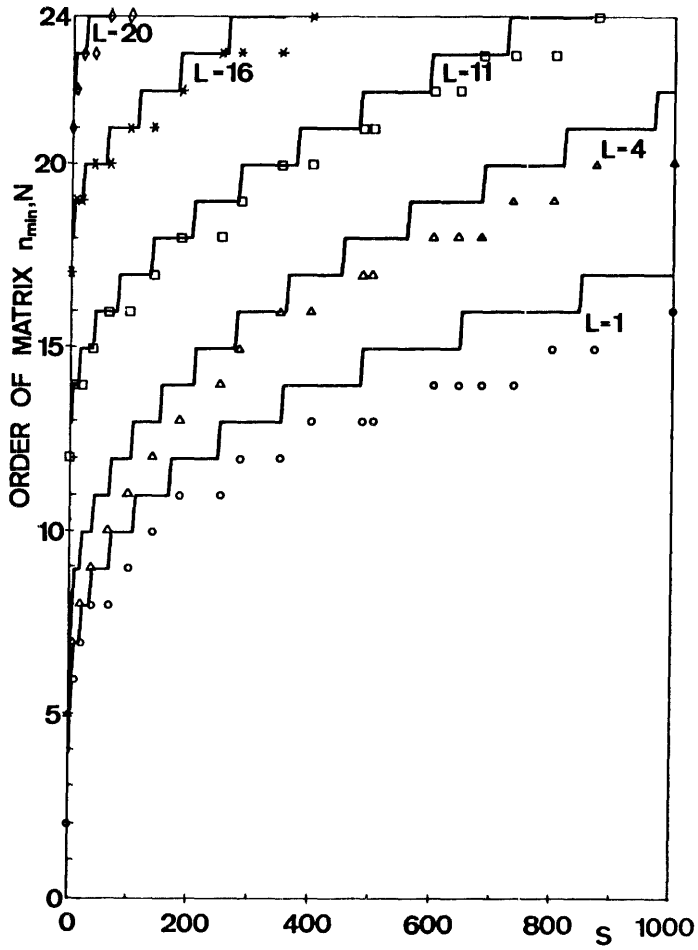


Fig. 1. Order of matrix (A, B, C, or D) necessary to give the first L characteristic values b correct to nine decimal places. Data points (○, △, □, *, and ◇) represent numerically found n_{min} in case of matrix D, while the full lines represent the function $N(S, L)$ according eq. (3). For clarity, only five values of L are presented.

turns out to be tridiagonal but unsymmetric. The only reason for unsymmetry is that $d'_{12} = S/4$ is unequal $d'_{21} = S/2$. Proper similarity transformation produces a symmetric tridiagonal matrix D with $d_{12} = d_{21} = S/(8^{1/2})$ and all other elements remain unchanged. This allows the use of one and the same library subroutine for determining the eigenvalues of all four matrices.

Next, one has to establish the minimum order $n_{\min}$ of the matrix A (or B, C, D) such as to yield a number L ($L < n_{\min}$) of characteristic values with given accuracy. In a modification of the program presented here, and using a trial and error method, we determined $n_{\min}$ as a function of both S and L for a b-accuracy of 9 decimal places. This was done for some discrete values of S and for integer L in the domain $0 \leq S \leq 1000$ and $0 \leq n_{\min} \leq 24$. The results are shown in Fig. 1 for some selected values of L. Then an analytic function $N(S, L)$ was developed which closely approximates $n_{\min}$, but ensures $N \geq n_{\min}$:

$$N(S, L) = \text{int}((0.17 + 2.1 \exp(-0.24\, L))S^{(0.77-5/(9.5+L)} + L + 2.8). \qquad (3)$$

The function $N(S, L)$ is presented in Fig. 1 as full line for the same parameters L as the numerically evaluated $n_{\min}(S, L)$. There are only minor differences in $n_{\min}$ required for A, B, C, and D, and eq. (3) has been determined as to fulfill $N \geq n_{\min}$ for all four matrices; therefore, it will be used for all four sets of characteristic values.

EXAMPLES

Find bo_1, $bo_3, \ldots$, bo_{29} for $S = 2$, also be_1, $be_3, \ldots$, be_7 for $S = 1000$; in addition, bo_2, $bo_4, \ldots$, bo_{38} for $S = 100$, and be_0, $be_2, \ldots$, be_{18} for $S = 0.1$. The results computed with a CDC CYBER/16 are given in Table II. Part of these results can be compared with tabulated values [2] and show agreement.

ACKNOWLEDGMENT

Thanks are due to the Computer Center of the Technische Universität Wien, which provided free computer time, and to R. Welser for his assistance.

Table II

S = 2.00	CHARACTERISTIC VALUES	S = 1000.00	CHARACTERISTIC VALUES
	BO 1 = 1.470654355		BE 1 = 93.599796781
	BO 3 = 10.013719839		BE 3 = 214.914687921
	BO 5 = 26.005209010		BE 5 = 331.837262869
	BO 7 = 50.002604266		BE 7 = 444.079857151
	BO 9 = 82.001562520		
	BO11 = 122.001041673		
	BO13 = 170.000744050		
	BO15 = 226.000558037		
	BO17 = 290.000434028		
	BO19 = 362.000347222		
	BO21 = 442.000284091	S = 100.00	CHARACTERISTIC VALUES
	BO23 = 530.000236742		
	BO25 = 626.000200321		BO 2 = 28.685139778
	BO27 = 730.000171703		BO 4 = 62.986489953
	BO29 = 842.000148809		BO 6 = 91.801071292
			BO 8 = 119.057988351
			BO10 = 153.225680042
			BO12 = 196.207674647
			BO14 = 247.611164916
S = .10	CHARACTERISTIC VALUES		BO16 = 307.229284862
			BO18 = 374.969344509
	BE 0 = .049687521		BO20 = 450.784185569
	BE 2 = 4.050260395		BO22 = 534.647547063
	BE 4 = 16.050020834		BO24 = 626.543802911
	BE 6 = 36.050008929		BO26 = 726.463163268
	BE 8 = 64.050004960		BO28 = 834.399234146
	BE10 = 100.050003157		BO30 = 950.347693024
	BE12 = 144.050002185		BO32 =1074.305531439
	BE14 = 196.050001603		BO34 =1206.270602586
	BE16 = 256.050001225		BO36 =1346.241340972
	BE18 = 324.050000967		BO38 =1494.216583111

REFERENCES

[Note. Reference [1] is not mentioned in the text.]

1. Goos, G., and Hartmanis, J. Eds., *Lecture Notes in Computer Science, vol. 6*, Springer, New York, 1976.
2. National Bureau of Standards, *Tables Relating to Mathieu Functions*. Cambridge U. Press, New York, 1951.

ALGORITHM

```
      SUBROUTINE CHARMA(KIND, S, L, NMAX, VAL, SUD, IA)                 ØØØØØØ1Ø
C SUBROUTINE CHARMA CALCULATES THE CHARACTERISTIC VALUES OF             ØØØØØØ2Ø
C MATHIEU-S DIFFERENTIAL EQUATION FOR ODD OR EVEN SOLUTIONS             ØØØØØØ3Ø
C WITH PERIODICITY PI OR 2*PI.                                          ØØØØØØ4Ø
C A LIBRARY SUBROUTINE HAS TO BE ATTACHED TO CALCULATE THE              ØØØØØØ5Ø
C EIGENVALUES OF A REAL SYMMETRIC TRIDIAGONAL MATRIX. IN THE            ØØØØØØ6Ø
C PRESENTED FORM, THIS IS THE EISPACK ROUTINE IMTQL1.                   ØØØØØØ7Ø
C INPUT..                                                               ØØØØØØ8Ø
C   KIND AN INTEGER CHARACTERIZING THE KIND OF CH. V.                   ØØØØØØ9Ø
C        IF KIND=1, THE CH. V. BO 1, BO 3,... FOR ODD SOLUTIONS         ØØØØ1ØØ
C           WITH PERIODICITY 2*PI ARE CALCULATED                        ØØØØ11Ø
C        IF KIND=2, THE CH. V. BE 1, BE 3,... FOR EVEN SOLUTIONS        ØØØØ12Ø
C           WITH PERIODICITY 2*PI ARE CALCULATED                        ØØØØ13Ø
C        IF KIND=3, THE CH. V. BO 2, BO 4,... FOR ODD SOLUTIONS         ØØØØ14Ø
C           WITH PERIODICITY PI ARE CALCULATED                          ØØØØ15Ø
C        IF KIND=4, THE CH. V. BE Ø, BE 2,... FOR EVEN SOLUTIONS        ØØØØ16Ø
C           WITH PERIODICITY PI ARE CALCULATED                          ØØØØ17Ø
C        CHARMA DOES NOT DESTROY KIND                                   ØØØØ18Ø
C   S    REAL NON-NEGATIVE VARIABLE, THE PARAMETER OF THE               ØØØØ19Ø
C        DIFFERENTIAL EQUATION. FOR S UP TO 1ØØØ AN ACCURACY OF         ØØØØ2ØØ
C        9 DECIMAL PLACES IS GUARANTEED FOR THE CHARACTERISTIC          ØØØØ21Ø
C        VALUES IF THE NUMBER OF THE CH. V. IS NOT TOO HIGH.            ØØØØ22Ø
C        (SEE FIG. 1 OF DESCRIPTION)                                    ØØØØ23Ø
C   L    INTEGER VARIABLE, THE NUMBER OF CHARACTERISTIC VALUES          ØØØØ24Ø
C        TO BE CALCULATED.  IT CAN BE NO LARGER THAN NMAX.              ØØØØ25Ø
C   NMAX INTEGER VARIABLE, MAXIMUM DIMENSION OF THE MATRIX USED         ØØØØ26Ø
C        FOR THE CALCULATION. TO MAKE USE OF THE FULL TESTED            ØØØØ27Ø
C        DOMAIN WITH AN ACCURACY OF 9 DECIMAL PLACES, IT SHOULD         ØØØØ28Ø
C        BE AT LEAST 24.                                                ØØØØ29Ø
C OUTPUT..                                                              ØØØØ3ØØ
C   VAL  REAL ONE-DIMENSIONAL ARRAY OF DIMENSION (NMAX).               ØØØØ31Ø
C        INITIALLY IT CONTAINS THE DIAGONAL ELEMENTS OF THE             ØØØØ32Ø
C        COEFFICIENT MATRIX.  ON EXIT, ITS FIRST L ELEMENTS WILL        ØØØØ33Ø
C        CONTAIN THE CH. V.                                             ØØØØ34Ø
C   IA   INTEGER VARIABLE USED AS A FAILURE INDICATOR. IF, ON           ØØØØ35Ø
C        EXIT, IA=Ø, NO FAILURE WAS DETECTED. IF IA=1, S WAS            ØØØØ36Ø
C        NEGATIVE.  IF IA=2, L WAS CHOSEN TOO BIG, REQUIRING A          ØØØØ37Ø
C        LARGER NMAX.  IF IA=3, THE LIBRARY SUBROUTINE IMTQL1           ØØØØ38Ø
C        DID NOT FIND ALL EIGENVALUES.  FOR IA=1, 2, OR 3, NO           ØØØØ39Ø
C        CHARACTERISTIC VALUES WERE CALCULATED.  IF IA=4,               ØØØØ4ØØ
C        S .GT. 1ØØØ.  IF IA=5, L WAS TOO LARGE (FOR THE GIVEN          ØØØØ41Ø
C        S), REQUIRING AN ORDER N OF THE COEFFICIENT MATRIX              ØØØØ42Ø
C        WHICH EXCEEDS THE TESTED DOMAIN (N .LE. 24).  FOR IA=4         ØØØØ43Ø
C        OR 5 THE CALCULATION WAS EXECUTED, BUT ACCURACY OF THE         ØØØØ44Ø
C        CHARACTERISTIC VALUES TO NINE DECIMAL PLACES IS NOT            ØØØØ45Ø
C        GUARANTEED.  IN THESE CASES, ACCURACY MAY BE CHECKED AS        ØØØØ46Ø
C        FOLLOWS.  IF L=L1 PRODUCES IA=4 OR 5, TAKE A VALUE L=L2,       ØØØØ47Ø
C        L2=L1+1, COMPUTE VAL (IA WILL AGAIN BE 4 OR 5) AND COMPUTE     ØØØØ48Ø
C        THE DIFFERENCE OF THE FIRST L1 MEMBERS OF BOTH SEQUENCES.      ØØØØ49Ø
C        IF THE ERROR TEST IS PASSED, ACCEPT THE ANSWER, IF NOT TAKE    ØØØØ5ØØ
C        A VALUE L=L3=L2+1, ETC...                                      ØØØØ51Ø
C OTHER PARAMETERS..                                                    ØØØØ52Ø
C  SUD   A ONE-DIMENSIONAL REAL ARRAY OF DIMENSION (NMAX),              ØØØØ53Ø
C        INITIALLY CONTAINING IN ITS POSITIONS (2), (3),..., (N)        ØØØØ54Ø
C        THE SUBDIAGONAL ELEMENTS OF THE COEFFICIENT MATRIX .           ØØØØ55Ø
C    N   INTEGER VARIABLE, THE ORDER OF THE COEFFICIENT MATRIX          ØØØØ56Ø
C   IB   AN INTEGER VARIABLE TO TEST THE SUCCESS OF IMTQL1. IF,         ØØØØ57Ø
C        ON EXIT, IB=Ø, IMTQL1 HAS DETERMINED ALL EIGENVALUES           ØØØØ58Ø
C        WITHIN 3Ø ITERATIONS.                                          ØØØØ59Ø
      REAL FL, S, SUD, VAL                                              ØØØØ6ØØ
      INTEGER I, IA, IB, IOUT, KIND, L, N, NMAX                         ØØØØ61Ø
      DIMENSION VAL(NMAX), SUD(NMAX)                                    ØØØØ62Ø
      IA = Ø                                                            ØØØØ63Ø
C TEST FOR NEGATIVE S                                                   ØØØØ64Ø
      IF (S.GE.Ø.ØE Ø) GO TO 1Ø                                         ØØØØ65Ø
```

```
      IA = 1                                                          ØØØØØ66Ø
      RETURN                                                          ØØØØØ67Ø
C TEST FOR S GREATER 1ØØØ. IF TRUE, CALCULATION IS CONTINUED          ØØØØØ68Ø
C BUT ACCURACY OF THE CHARACTERSTIC VALUES TO NINE DECIMAL            ØØØØØ69Ø
C PLACES CANNOT BE GUARANTEED.  SEE COMMENT ON IA IN OUTPUT LIST      ØØØØØ7ØØ
C FOR IMPROVEMENT.                                                    ØØØØØ71Ø
   1Ø IF (S.LE.1ØØØ.ØE Ø) GO TO 2Ø                                    ØØØØØ72Ø
      IA = 4                                                          ØØØØØ73Ø
C DETERMINE NECESSARY ORDER OF MATRIX TO ACHIEVE AN ACCURACY          ØØØØØ74Ø
C OF 9 DECIMAL PLACES FOR THE CHARACTERISTIC VALUES.                  ØØØØØ75Ø
   2Ø FL = FLOAT(L)                                                   ØØØØØ76Ø
      N = INT((Ø.17E Ø+2.1E Ø*EXP(-Ø.24E Ø*FL))*S**(Ø.77E Ø-5.ØE Ø/   ØØØØØ77Ø
     * (9.5E Ø+FL))+FL+2.8E Ø)                                        ØØØØØ78Ø
C TEST FOR SUFFICIENT LARGE NMAX                                      ØØØØØ79Ø
      IF (N.LE.NMAX) GO TO 3Ø                                         ØØØØØ8ØØ
      IA = 2                                                          ØØØØØ81Ø
      RETURN                                                          ØØØØØ82Ø
C TEST WHETHER N IS WITHIN TESTED DOMAIN FOR WHICH ACCURACY TO 9      ØØØØØ83Ø
C DECIMAL PLACES IS GUARANTEED. IF NOT, CALCULATION IS CONTINUED      ØØØØØ84Ø
C BUT ACCURACY OF THE CHARACTERSTIC VALUES TO NINE DECIMAL            ØØØØØ85Ø
C PLACES CANNOT BE GUARANTEED.  SEE COMMENT ON IA IN OUTPUT LIST      ØØØØØ86Ø
C FOR IMPROVEMENT.                                                    ØØØØØ87Ø
   3Ø IF (N.LE.24) GO TO 4Ø                                           ØØØØØ88Ø
      IA = 5                                                          ØØØØØ89Ø
C BRANCH ACCORDING TO DESIRED SOLUTION:                               ØØØØØ9ØØ
C IF KIND=1, USE MATRIX CALLED A IN THE DESCRIPTION                   ØØØØØ91Ø
C IF KIND=2, USE MATRIX CALLED B IN THE DESCRIPTION                   ØØØØØ92Ø
C IF KIND=3, USE MATRIX CALLED C IN THE DESCRIPTION                   ØØØØØ93Ø
C IF KIND=4, USE MATRIX CALLED D IN THE DESCRIPTION                   ØØØØØ94Ø
   4Ø GO TO (5Ø, 6Ø, 9Ø, 11Ø), KIND                                   ØØØØØ95Ø
C STORE DIAGONAL ELEMENTS OF THE COEFFICIENT MATRIX IN VAL AND        ØØØØØ96Ø
C SUBDIAGONAL ELEMENTS IN SUD(2), SUD(3),..., SUD(N).                 ØØØØØ97Ø
   5Ø VAL(1) = 1.ØE Ø - S/4.ØE Ø                                      ØØØØØ98Ø
      GO TO 7Ø                                                        ØØØØØ99Ø
   6Ø VAL(1) = 1.ØE Ø + S/4.ØE Ø                                      ØØØØ1ØØØ
   7Ø DO 8Ø I=2,N                                                     ØØØØ1Ø1Ø
        VAL(I) = FLOAT((2*I-1)**2)                                    ØØØØ1Ø2Ø
        SUD(I) = S/4.ØE Ø                                             ØØØØ1Ø3Ø
   8Ø CONTINUE                                                        ØØØØ1Ø4Ø
      GO TO 13Ø                                                       ØØØØ1Ø5Ø
   9Ø VAL(1) = 4.ØE Ø                                                 ØØØØ1Ø6Ø
      DO 1ØØ I=2,N                                                    ØØØØ1Ø7Ø
        VAL(I) = FLOAT((2*I)**2)                                      ØØØØ1Ø8Ø
        SUD(I) = S/4.ØE Ø                                             ØØØØ1Ø9Ø
  1ØØ CONTINUE                                                        ØØØØ11ØØ
      GO TO 13Ø                                                       ØØØØ111Ø
  11Ø VAL(1) = Ø.ØE Ø                                                 ØØØØ112Ø
      VAL(2) = 4.ØE Ø                                                 ØØØØ113Ø
      SUD(2) = S/SQRT(8.ØE Ø)                                         ØØØØ114Ø
      DO 12Ø I=3,N                                                    ØØØØ115Ø
        VAL(I) = FLOAT((2*(I-1))**2)                                  ØØØØ116Ø
        SUD(I) = S/4.ØE Ø                                             ØØØØ117Ø
  12Ø CONTINUE                                                        ØØØØ118Ø
  13Ø CALL IMTQL1(N, VAL, SUD, IB)                                    ØØØØ119Ø
C TEST FOR SUCCESSFUL IMTQL1                                          ØØØØ12ØØ
      IF (IB.EQ.Ø) GO TO 14Ø                                          ØØØØ121Ø
      IA = 3                                                          ØØØØ122Ø
      RETURN                                                          ØØØØ123Ø
C ADD S/2 TO THE EIGENVALUES OF THE MATRIX TO GET THE CH. V.          ØØØØ124Ø
  14Ø DO 15Ø I=1,L                                                    ØØØØ125Ø
        VAL(I) = VAL(I) + S/2.ØE Ø                                    ØØØØ126Ø
  15Ø CONTINUE                                                        ØØØØ127Ø
      RETURN                                                          ØØØØ128Ø
      END                                                             ØØØØ129Ø


C      PROGRAM TECHV (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)          ØØØØ13ØØ
C THIS IS A PROGRAM TO TEST THE SUBROUTINE CHARMA WHICH CALCULATES THE ØØØØ131Ø
C CHARACTERISTIC VALUES OF MATHIEU'S DIFFERENTIAL EQUATION FOR ODD OR  ØØØØ132Ø
C EVEN SOLUTIONS WITH PERIODICITY PI OR 2*PI.                         ØØØØ133Ø
C INPUT..                                                             ØØØØ134Ø
C   S     A REAL VARIABLE, THE PARAMETER OF THE DIFFERENTIAL EQUATION ØØØØ135Ø
C   L     AN INTEGER, THE NUMBER OF CHARACTERISTIC VALUES TO BE       ØØØØ136Ø
C         CALCULATED                                                  ØØØØ137Ø
C OUTPUT..                                                            ØØØØ138Ø
C   IA    AN INTEGER USED AS A FAILURE INDICATOR. IF, ON EXIT, IA=Ø, NO ØØØØ139Ø
C         FAILURE WAS DETECTED. IF IA=1 OR IA=2 OR IA=3, A FAILURE WAS ØØØØ14ØØ
```

```
C         REGISTERED AND NO CHARACTERISTIC VALUES WERE CALCULATED.      ØØØØ141Ø
C   VAL   A ONE-DIMENSIONAL ARRAY USED BY CHARMA AND BY A LIBRARY       ØØØØ142Ø
C         SUBROUTINE. THE DIMENSION OF VAL SHOULD BE EQUAL OR GREATER   ØØØØ143Ø
C         THAN 24 TO ACHIEVE FULL ACCURACY OVER THE WHOLE TESTED DOMAINEØØØØ144Ø
C         OF CHARMA. ON EXIT, VAL CONTAINS THE CHARACTERISTIC VALUES.   ØØØØ145Ø
C INDE    INTEGER VARIABLE, THE INDEX OF THE CHARACTERISTIC VALUES      ØØØØ146Ø
C KIND    AN INTEGER CHARACTERIZING THE KIND OF CHARACTERISTIC VALUE    ØØØØ147Ø
C         TO BE CALCULATED                                              ØØØØ148Ø
C SUD     A ONE-DIMENSIONAL ARRAY USED BY CHARMA AND BY A LIBRARY       ØØØØ149Ø
C         SUBROUTINE. THE DIMENSION OF SUD SHOULD BE EQUAL OR GREATER   ØØØØ15ØØ
C         THAN 24 TO ACHIEVE FULL ACCURACY OVER THE WHOLE TESTED DOMAINEØØØØ151Ø
C         OF CHARMA.                                                    ØØØØ152Ø
C THE FOLLOWING DATA CARDS HAVE BEEN USED FOR THIS TEST PROGRAM (S=-1.  ØØØØ153Ø
C SIGNALS END OF DATA)                                                  ØØØØ154Ø
C   2.    15                                                            ØØØØ155Ø
C1ØØ1.     4                                                            ØØØØ156Ø
C   Ø.Ø1  25                                                            ØØØØ157Ø
C  1Ø.    27                                                            ØØØØ158Ø
C  -6.     3                                                            ØØØØ159Ø
C  -1.                                                                  ØØØØ16ØØ
      DIMENSION VAL(28), SUD(28)                                        ØØØØ161Ø
   1Ø READ (5,99999) S, L                                               ØØØØ162Ø
      IF (S.EQ.-1.) GO TO 8Ø                                            ØØØØ163Ø
C TEST SUBROUTINE FOR ALL FOUR KIND OF SOLUTIONS                        ØØØØ164Ø
      DO 7Ø KIND=1,4                                                    ØØØØ165Ø
        CALL CHARMA(KIND, S, L, 28, VAL, SUD, IA)                       ØØØØ166Ø
        IF (IA.GT.Ø) WRITE (6,99993) S, L, IA                           ØØØØ167Ø
        IF ((IA.GT.Ø) .AND. (IA.LT.4)) GO TO 1Ø                         ØØØØ168Ø
        WRITE (6,99998) S                                               ØØØØ169Ø
C PRINT THE FIRST L ELEMENTS OF VAL, WHICH ARE THE CHARACTERISTIC VALUESØØØØ17ØØ
        DO 6Ø I=1,L                                                     ØØØØ171Ø
C GIVE NAMES AND INDICES OF CH. V. ACCORDING KIND OF SOLUTION           ØØØØ172Ø
          GO TO (2Ø, 3Ø, 4Ø, 5Ø), KIND                                 ØØØØ173Ø
   2Ø     INDE = 2*I - 1                                                ØØØØ174Ø
          WRITE (6,99997) INDE, VAL(I)                                  ØØØØ175Ø
          GO TO 6Ø                                                      ØØØØ176Ø
   3Ø     INDE = 2*I - 1                                                ØØØØ177Ø
          WRITE (6,99996) INDE, VAL(I)                                  ØØØØ178Ø
          GO TO 6Ø                                                      ØØØØ179Ø
   4Ø     INDE = 2*I                                                    ØØØØ18ØØ
          WRITE (6,99995) INDE, VAL(I)                                  ØØØØ181Ø
          GO TO 6Ø                                                      ØØØØ182Ø
   5Ø     INDE = 2*I - 2                                                ØØØØ183Ø
          WRITE (6,99994) INDE, VAL(I)                                  ØØØØ184Ø
   6Ø   CONTINUE                                                        ØØØØ185Ø
   7Ø CONTINUE                                                          ØØØØ186Ø
      GO TO 1Ø                                                          ØØØØ187Ø
   8Ø CONTINUE                                                          ØØØØ188Ø
      STOP                                                              ØØØØ189Ø
99999 FORMAT (F8.2, 2X, I2)                                             ØØØØ19ØØ
99998 FORMAT (1HØ, 17X, 3HS =, F8.2, 2X, 19HCHARACTERISTIC VALU,        ØØØØ191Ø
     * 2HES/)                                                           ØØØØ192Ø
99997 FORMAT (31X, 2HBO, I2, 2H =, F14.9)                               ØØØØ193Ø
99996 FORMAT (31X, 2HBE, I2, 2H =, F14.9)                               ØØØØ194Ø
99995 FORMAT (31X, 2HBO, I2, 2H =, F14.9)                               ØØØØ195Ø
99994 FORMAT (31X, 2HBE, I2, 2H =, F14.9)                               ØØØØ196Ø
99993 FORMAT (1HØ, 3ØX, 1ØHDATA - S =, F8.2/38X, 3HL =,                 ØØØØ197Ø
     * I3//26X, 15HERROR FLAG IA =, I3//)                               ØØØØ198Ø
      END                                                               ØØØØ199Ø
C                                                                       ØØØØ2ØØØ
C     ------------------------------------------------------------------ØØØØ2Ø1Ø
C                                                                       ØØØØ2Ø2Ø
      SUBROUTINE IMTQL1(N,D,E,IERR)                                     ØØØØ2Ø3Ø
C                                                                       ØØØØ2Ø4Ø
      INTEGER I,J,L,M,N,II,MML,IERR                                     ØØØØ2Ø5Ø
      REAL D(N),E(N)                                                    ØØØØ2Ø6Ø
      REAL B,C,F,G,P,R,S,MACHEP                                         ØØØØ2Ø7Ø
C     REAL SQRT,ABS,SIGN                                                ØØØØ2Ø8Ø
C                                                                       ØØØØ2Ø9Ø
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE IMTQL1,   ØØØØ21ØØ
C     NUM. MATH. 12, 377-383(1968) BY MARTIN AND WILKINSON,             ØØØØ211Ø
C     AS MODIFIED IN NUM. MATH. 15, 45Ø(197Ø) BY DUBRULLE.              ØØØØ212Ø
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 241-248(1971).   ØØØØ213Ø
C                                                                       ØØØØ214Ø
C     THIS SUBROUTINE FINDS THE EIGENVALUES OF A SYMMETRIC              ØØØØ215Ø
C     TRIDIAGONAL MATRIX BY THE IMPLICIT QL METHOD.                     ØØØØ216Ø
```

```
C                                                                       ØØØØ217Ø
C     ON INPUT-                                                         ØØØØ218Ø
C                                                                       ØØØØ219Ø
C        N IS THE ORDER OF THE MATRIX,                                  ØØØØ22ØØ
C                                                                       ØØØØ221Ø
C        D CONTAINS THE DIAGONAL ELEMENTS OF THE INPUT MATRIX,          ØØØØ222Ø
C                                                                       ØØØØ223Ø
C        E CONTAINS THE SUBDIAGONAL ELEMENTS OF THE INPUT MATRIX        ØØØØ224Ø
C          IN ITS LAST N-1 POSITIONS.  E(1) IS ARBITRARY.               ØØØØ225Ø
C                                                                       ØØØØ226Ø
C      ON OUTPUT-                                                       ØØØØ227Ø
C                                                                       ØØØØ228Ø
C        D CONTAINS THE EIGENVALUES IN ASCENDING ORDER.  IF AN          ØØØØ229Ø
C          ERROR EXIT IS MADE, THE EIGENVALUES ARE CORRECT AND          ØØØØ23ØØ
C          ORDERED FOR INDICES 1,2,...IERR-1, BUT MAY NOT BE            ØØØØ231Ø
C          THE SMALLEST EIGENVALUES,                                    ØØØØ232Ø
C                                                                       ØØØØ233Ø
C        E HAS BEEN DESTROYED,                                          ØØØØ234Ø
C                                                                       ØØØØ235Ø
C        IERR IS SET TO                                                 ØØØØ236Ø
C          ZERO       FOR NORMAL RETURN,                               ØØØØ237Ø
C          J          IF THE J-TH EIGENVALUE HAS NOT BEEN              ØØØØ238Ø
C                     DETERMINED AFTER 3Ø ITERATIONS.                  ØØØØ239Ø
C                                                                       ØØØØ24ØØ
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO B. S. GARBOW,        ØØØØ241Ø
C     APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY         ØØØØ242Ø
C                                                                       ØØØØ243Ø
C     ------------------------------------------------------------------ØØØØ244Ø
C                                                                       ØØØØ245Ø
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING     ØØØØ246Ø
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.   ØØØØ247Ø
C                                                                       ØØØØ248Ø
C                **********                                             ØØØØ249Ø
      MACHEP = 2.**(-26)                                                ØØØØ25ØØ
C                                                                       ØØØØ251Ø
      IERR = Ø                                                          ØØØØ252Ø
      IF (N .EQ. 1) GO TO 1ØØ1                                          ØØØØ253Ø
C                                                                       ØØØØ254Ø
      DO 1ØØ I = 2, N                                                   ØØØØ255Ø
  1ØØ E(I-1) = E(I)                                                     ØØØØ256Ø
C                                                                       ØØØØ257Ø
      E(N) = Ø.Ø                                                        ØØØØ258Ø
C                                                                       ØØØØ259Ø
      DO 29Ø L = 1, N                                                   ØØØØ26ØØ
         J = Ø                                                          ØØØØ261Ø
C     ********** LOOK FOR SMALL SUB-DIAGONAL ELEMENT **********        ØØØØ262Ø
  1Ø5    DO 11Ø M = L, N                                               ØØØØ263Ø
            IF (M .EQ. N) GO TO 12Ø                                     ØØØØ264Ø
            IF (ABS(E(M)) .LE. MACHEP * (ABS(D(M)) + ABS(D(M+1))))      ØØØØ265Ø
     X         GO TO 12Ø                                               ØØØØ266Ø
  11Ø    CONTINUE                                                      ØØØØ267Ø
C                                                                       ØØØØ268Ø
  12Ø    P = D(L)                                                      ØØØØ269Ø
         IF (M .EQ. L) GO TO 215                                        ØØØØ27ØØ
         IF (J .EQ. 3Ø) GO TO 1ØØØ                                      ØØØØ271Ø
         J = J + 1                                                      ØØØØ272Ø
C     ********** FORM SHIFT **********                                 ØØØØ273Ø
         G = (D(L+1) - P) / (2.Ø * E(L))                                ØØØØ274Ø
         R = SQRT(G*G+1.Ø)                                              ØØØØ275Ø
         G = D(M) - P + E(L) / (G + SIGN(R,G))                          ØØØØ276Ø
         S = 1.Ø                                                        ØØØØ277Ø
         C = 1.Ø                                                        ØØØØ278Ø
         P = Ø.Ø                                                        ØØØØ279Ø
         MML = M - L                                                    ØØØØ28ØØ
C     ********** FOR I=M-1 STEP -1 UNTIL L DO -- **********            ØØØØ281Ø
         DO 2ØØ II = 1, MML                                             ØØØØ282Ø
            I = M - II                                                  ØØØØ283Ø
            F = S * E(I)                                                ØØØØ284Ø
            B = C * E(I)                                                ØØØØ285Ø
            IF (ABS(F) .LT. ABS(G)) GO TO 15Ø                           ØØØØ286Ø
            C = G / F                                                   ØØØØ287Ø
            R = SQRT(C*C+1.Ø)                                           ØØØØ288Ø
            E(I+1) = F * R                                              ØØØØ289Ø
            S = 1.Ø / R                                                 ØØØØ29ØØ
            C = C * S                                                   ØØØØ291Ø
            GO TO 16Ø                                                   ØØØØ292Ø
```

```
  15Ø       S = F / G                                                    ØØØØ293Ø
            R = SQRT(S*S+1.Ø)                                            ØØØØ294Ø
            E(I+1) = G * R                                               ØØØØ295Ø
            C = 1.Ø / R                                                  ØØØØ296Ø
            S = S * C                                                    ØØØØ297Ø
  16Ø       G = D(I+1) - P                                               ØØØØ298Ø
            R = (D(I) - G) * S + 2.Ø * C * B                             ØØØØ299Ø
            P = S * R                                                    ØØØØ3ØØØ
            D(I+1) = G + P                                               ØØØØ3Ø1Ø
            G = C * R - B                                                ØØØØ3Ø2Ø
  2ØØ     CONTINUE                                                       ØØØØ3Ø3Ø
C                                                                        ØØØØ3Ø4Ø
         D(L) = D(L) - P                                                 ØØØØ3Ø5Ø
         E(L) = G                                                        ØØØØ3Ø6Ø
         E(M) = Ø.Ø                                                      ØØØØ3Ø7Ø
         GO TO 1Ø5                                                       ØØØØ3Ø8Ø
C     ********** ORDER EIGENVALUES **********                           ØØØØ3Ø9Ø
  215    IF (L .EQ. 1) GO TO 25Ø                                         ØØØØ31ØØ
C     ********** FOR I=L STEP -1 UNTIL 2 DO -- **********               ØØØØ311Ø
         DO 23Ø II = 2, L                                                ØØØØ312Ø
            I = L + 2 - II                                               ØØØØ313Ø
            IF (P .GE. D(I-1)) GO TO 27Ø                                 ØØØØ314Ø
            D(I) = D(I-1)                                                ØØØØ315Ø
  23Ø    CONTINUE                                                        ØØØØ316Ø
C                                                                        ØØØØ317Ø
  25Ø    I = 1                                                           ØØØØ318Ø
  27Ø    D(I) = P                                                        ØØØØ319Ø
  29Ø CONTINUE                                                           ØØØØ32ØØ
C                                                                        ØØØØ321Ø
      GO TO 1ØØ1                                                         ØØØØ322Ø
C     ********** SET ERROR -- NO CONVERGENCE TO AN                       ØØØØ323Ø
C                EIGENVALUE AFTER 3Ø ITERATIONS **********               ØØØØ324Ø
 1ØØØ IERR = L                                                           ØØØØ325Ø
 1ØØ1 RETURN                                                             ØØØØ326Ø
C     ********** LAST CARD OF IMTQL1 **********                         ØØØØ327Ø
      END                                                                ØØØØ328Ø
```

538-P 1-0

ALGORITHM 538
Eigenvectors and Eigenvalues of Real Generalized Symmetric Matrices by Simultaneous Iteration [F2]

PAUL J. NIKOLAI
U.S. Air Force Flight Dynamics Laboratory

Key Words: and Phrases: eigenvalue, eigenvector, sparse matrix, diagonable matrix, simultaneous iteration, Fortran program
CR Categories: 4.6, 5.14
Language: Fortran

DESCRIPTION

The program presented here is an implementation of the simultaneous iteration algorithm [2] for calculating the eigenvalues largest in magnitude and corresponding eigenvectors of a real matrix symmetric relative to a prescribed inner product. Let $\mathrm{ip}(n, w, z)$ denote an inner product in the space of real column n-tuples and let the real n-square matrix C satisfy $\mathrm{ip}(n, Cw, z) = \mathrm{ip}(n, w, Cz)$. Then C is *symmetric relative to* ip, and if the n-square positive definite matrix B satisfies $\mathrm{ip}(n, w, z) = w^{\mathrm{T}}Bz$ then C is *B-symmetric*. The equation $BC = C^{\mathrm{T}}B$ characterizes the B-symmetry of C. Given an optional set of p initial approximate eigenvectors of a real n-square B-symmetric matrix C corresponding to p eigenvalues of C largest in magnitude, the program calculates em eigenvalues and em corresponding eigenvectors, $0 \leqq em < p \leqq n$, to a precision dependent on the structure of C and on a prescribed tolerance eps. The matrix B is presented to the program as an independently prepared real function subprogram which calculates $\mathrm{ip}(n, w, z) = w^{\mathrm{T}}Bz$ given column n-vectors w and z. The matrix C is presented as an independently prepared subroutine subprogram $\mathrm{op}(n, z, w)$ which when given an n-vector z computes its image $w = Cz$. The program is an outgrowth of a literal Fortran translation [5] of the Algol procedure *ritzit* [8] to which it is substantially equivalent when $C = C^{\mathrm{T}}$ and $\mathrm{ip}(n, w, z) = w^{\mathrm{T}}z$, the standard inner product. But depending on the choice of B and C, the present program enables the direct treatment of a wide variety of symmetric eigenproblems.

Let $A = A^{\mathrm{T}}$ and $B = B^{\mathrm{T}}$ denote n-square real matrices and let σ be real. If B is positive definite then the matrix $C = B^{-1}(A - \sigma B)$ is B-symmetric, and the program computes eigenvalues farthest from σ of the eigenproblem $Au = \lambda Bu$ and corresponding eigenvectors. Implementation of $\mathrm{op}(n, z, w)$ here consists in providing for the appropriate solution for w of the linear system $Bw = (A - \sigma B)z$. Alternatively, selection of op to solve the system $(A - \sigma B)w = Bz$ for w enables

Received 13 April 1977 and 23 August 1977.

Author's address: Department of the Air Force, Applied Mathematics Group, Analysis and Optimization Branch, Structural Mechanics Division, Wright-Patterson Air Force Base, Ohio 45433.

the calculation by simultaneous inverse iteration of the eigenvalues nearest to σ and their eigenvectors. Implications for large sparse systems for which the Cholesky factorization [6] of B is impractical are clear. The user may wish to supplement the following outline of the operation of the program by consulting the description of the Algol procedure *ritzit* in [8] or [10] as well as a review of the mathematical foundations of simultaneous iteration in [4] and [7].

Let the eigenvalues $d_1, \ldots, d_p, d_{p+1}, \ldots, d_n$ of C be arranged in order of descending absolute value and let E_p denote the direct sum of the distinct eigenspaces corresponding to $d_1, \ldots, d_p$. Let X_0 denote an n-by-p matrix having a p-dimensional column space not orthogonal relative to ip to any eigenvector in E_p. Simultaneous iteration is based on the observation that if $|d_p| > |d_{p+1}|$, the columns of the matrix $X_{k+m} = C^m X_k$ tend to a basis of E_p as $ks = k + m$ increases. But in practice all of the columns of X_{ks} tend toward the eigenspace E_1 causing loss of information concerning the residual eigenvectors. To counter this tendency, set

$$X_{k+m} = C^m X_k R_{k+m}^{-1} \tag{1}$$

where the p-square upper triangular matrix R_{ks} is constructed together with X_{ks} by the Gram-Schmidt process to render the columns of X_{ks} orthonormal relative to ip. Now the ith column vector of X_{ks} converges to the ith eigenvector of C at a rate proportional to $\max_{2 \leqq i \leqq p}(|d_i/d_{i-1}|, |d_{i+1}/d_i|)$. Clearly this convergence will be delayed in the presence of eigenvalue clustering. But if $|d_p|/|d_{p+1}|$ is not too small, the column space of X_{ks} will contain a good approximation to the ith eigenvector even when ks is small.

In order to recover this approximation, a modified Rayleigh-Ritz process is employed. Let Q_{ks} denote an orthogonal matrix which diagonalizes the p-square symmetric matrix $R_{ks}R_{ks}^{\mathrm{T}}$. Then the ith column vector of

$$X_{k+1} = CX_k R_{k+1}^{-1} Q_{k+1} \tag{2}$$

converges to the ith eigenvector of C at a rate proportional to $|d_{p+1}/d_i|$ while the entries of the diagonal matrix computed with Q_{ks} and properly ordered offer close approximations to $d_1^2, \ldots, d_p^2$. The true signed eigenvalues need only be computed at termination by diagonalizing the leading $(p-1)$-square principal submatrix of $X_{ks}^T BCX_{ks}$, the eigenproblem for C projected on E_{p-1} relative to ip.

The program determines a strategy for employing the devices (1) and (2) based on the distribution of the leading p eigenvalues of C upon which the convergence rate ultimately depends. The selection of values m in eq. (1) is particularly important in this regard in that $C^m X_k$ is replaced by the mth Chebychev polynomial on the interval $[-e, e]$ evaluated by a special three-term recurrence relation and permitting accelerated convergence when values of m are continually large; e is the current value of d_p. As a result the convergence quotient lies between $|d_p/d_{em}|$ and $\exp(-\text{arc cosh}\,|d_{em}/d_p|)$. It is nearer to the first value if $|d_1/d_{em}|$ is large and nearer to the second if the latter quotient is close to 1.

As the iteration proceeds through a maximum of $|km|$ iteration steps (km is a program parameter), acceptance tests for the eigenvalues and eigenvectors are conducted following each of the Rayleigh-Ritz steps (2). As soon as the relative increase of $|d_{h+1}|$ is smaller than eps/10, then d_{h+1} is accepted and h, the number of previously accepted eigenvalues, is increased by 1. Eigenvectors are accepted in groups of one or more corresponding to clusters of accepted eigenvalues nearly equal in magnitude. If g eigenvectors have already been accepted, let $d_{g+1}, \ldots, d_l$ denote such a cluster. For all j, $g + 1 \leqq j \leqq l$, denote by y_j the projection relative to ip of the image Cx_j of the tth column x_j of X_{ks} on the linear closure of $x_1, \ldots, x_l$. Set $f_i = \max_j \| Cx_j - y_j \| / \| Cx_j \|$ for $i = g + 1, \ldots, l$ where the indicated norm is the Euclidean norm or 2-norm relative to ip. If $|d_l| f_l/(|d_l| - e)$ is smaller than eps then all the $x_j, j = g + 1, \ldots, l$, are accepted as eigenvectors and g is increased to l. The error quantities f_i are systematically discounted in accordance with the convergence properties of the algorithm to permit convergence in the presence of excessive roundoff error or in case the parameter eps is prescribed unrealistically

small. Having determined g eigenvectors, the iteration continues with $p - g$ remaining columns of X_{ks} until either *em* eigenvectors have been calcuated or $|km|$ has been exceeded. The program may reduce *em* if it detects either no progress in convergence of eigenvectors corresponding to smaller eigenvalues or lack of stability in the behavior of larger eigenvalues.

We list the principal differences between *ritzit* and the program given here.

(1) The procedure *inprod* for calculating standard inner products was removed and the procedure ip was introduced where appropriate.

(2) The procedure *jacobi* for calculating the solutions of the eigenproblem for $(p - g)$-square and $(p - 1)$-square symmetric matrices was replaced by calls to the EISPACK [9] subroutines TRED2 and IMTQL2, primarily to save space.

(3) The procedure *random* for calculating random column n-vectors of the matrix X_{ks} was replaced by in-line code which references a Fortran function RANF. RANF returns uniformly distributed random REAL values from the interval (0, 1), one per function reference, given any one argument of any type. It is provided by the user.

(4) The procedure *orthog* to perform Gram-Schmidt orthogonalization of the columns of X_{ks} was replaced by internally linked in-line code. In attempting to control potential underflow within *orthog* in a machine independent fashion, *ritzit* calculates the machine precision *mc* but assumes in usage that out-of-range values underflow gracefully to zero, a machine dependent characteristic. The present program utilizes a single REAL machine dependent constant MT, the ratio of the smallest Fortran representable positive value to the machine precision, to test for this condition and upon its detection to take appropriate measures.

(5) In its Algol implementation *ritzit* requires approximately $(p + 3)n + 2p^2 + 5p$ storage locations in excess of those required by the program. Economies resulting in part from (2) above [3] have reduced this requirement to $(p + 2)n + p^2 + 4p$ in the program presented here. All working storage is confined to a single array of $2n + p^2 + 3p$ locations.

(6) The value of *km* as an input parameter, set to $|km|$ during program execution, is finally replaced by the value of *ks* as an output parameter, the number of iteration steps used in the calculation of *em* eigenvectors.

(7) The program given here retains unchanged the reference to a user supplied procedure inf as a window on program execution. However, the one variable involving eps is periodically redefined to enable effect control of eps from inf or from ip or op should this prove desirable.

The testing procedures developed for the present Fortran program parallel its evolution from a research tool, which conformed closely to its Algol parent, to present form. Early testing was concentrated on duplicating the tests furnished with *ritzit* and in eliminating errors in interpretation and translation of the Algol code. This was done for the most part on the CDC 6600 using Fortran Extended, Version 3, under the SCOPE 3.3 operating system. Upon completion of this first phase, the resulting program was distributed as SUBROUTINE RITZIT with a locally developed library of Fortran linear algebra routines [5]. This same program served as a basis of *ritzit* translations for IBM 360/370 processors [1, 3] whose preparation uncovered several bugs in the RITZIT code and suggested worthwhile modifications. A second phase of testing involved the development of a package of auxiliary Fortran programs for use with SUBROUTINE RITZIT to solve the eigenvalue problem $Au = \lambda Bu$ through methods depending on Cholesky factorization where A and B may either be full matrices or sparse and banded. This phase was conducted on the CDC 6600 using Fortran Extended, Version 4, under SCOPE 3.4.

Systematic testing of the present program, SUBROUTINE SIMITZ, has been accomplished in part with the aid of driver program TESTB which generates a symmetric band matrix A and a lower triangular band matrix T of prescribed order n and bandwidths whose relevant entries are randomly generated integer values from a prescribed interval. The band matrix B is TT^{T}, and the program

calculates the maximal eigenvalues of $Au = \lambda Bu$. For the sequence of values of p, $p = 2, \ldots,$ min([$n/5$], 10), TESTB exercises SIMITZ for successive values of em, $em = 1, \ldots, p - 1$. For each value of i, $i = 1, \ldots, em$, TESTB computes the residuals $Ax_i - d_iBx_i$ and their Euclidean norms relative to the standard inner product. Each norm is normalized by the difference $|d_1| - e$, and for each value of em the quantity $\max_{1 \le i \le em} \|Ax_i - d_iBx_i\|/(|d_1| - e)$, the value k of i for which the maximum occurs, and the corresponding geometric mean with unit weights are listed. Also listed are the relevant nonzero diagonals of A and T and the final eigenvalues computed for $em = p - 1$.

Figure 1 shows an output listing from the executable program TESTB on the CDC 6600 under the NOS/BE operating system and Fortran Extended, Version 4. Here A and B are of order 30 and each of bandwidth 7 having relevant entries between −99 and +99. Listed are the main diagonal and the three adjacent lower diagonals of T and A beginning with the entries in the first column. Here eps = 10^{-10} and km = 100. Note how the relative nearness in magnitude of the first three eigenvalues inhibits the convergence of the second eigenvector when $p = 3$ and $em = 2$ resulting in acceptance of the first eigenvector only. The fourth eigenvalue, however, is in absolute value far enough away from this cluster to permit successful convergence when $p = 4$ and $em = 1$, 2, and 3. This phenomenon points to a procedure for pursuing a solution when p is initially chosen too small. SIMITZ may be reentered with X containing the approximate eigenvectors calculated for the smaller value of p as initial approximations for use with p increased in size. Significant processor time may often be saved in this fashion.

ADJACENT NON-ZERO LOWER DIAGONALS OF T, TT' = B

53	86	78	60	85	99	73	64	80	88	93	93	57	52	83
74	99	88	87	52	64	86	95	66	86	55	95	61	76	70
-26	22	49	65	-30	38	82	-9	-87	56	18	95	93	-46	-31
1	90	-44	75	-6	-49	99	90	-62	16	2	-40	-67	14	
-25	6	54	32	-73	-35	-1	-46	6	-13	-77	77	-51	-85	31
-24	88	-66	-18	-41	5	80	74	-80	-2	-6	78	-97		
-76	19	-26	-5	97	70	6	-95	45	-30	-12	-65	-92	9	39
97	-19	26	86	-48	-52	-31	49	-8	-85	5	20			

ADJACENT NON-ZERO LOWER DIAGONALS OF A = A'

-3	50	-55	-73	-47	88	-33	-28	-77	-40	96	78	9	-54	-39
10	95	-53	-80	0	52	44	0	-50	-94	-22	29	-36	-4	36
0	65	59	-29	-84	-4	41	-90	72	-65	74	36	82	-86	-5
22	56	94	9	-47	84	-97	-1	57	-22	-1	-55	-15	51	
62	5	-41	-95	-47	-91	14	84	-83	69	43	-27	61	-79	14
21	85	-5	-61	-16	-15	90	-44	-22	-66	50	-34	-97		
-17	-94	78	-95	-89	-31	-43	52	-64	-70	71	95	-69	-12	87
91	-93	6	39	66	32	-20	92	-11	0	-98	-12			

P	EM(IN)	EM(OUT)	KS	K	MAX RESIDUAL	MEAN RESIDUAL	TIME(SECS)
2	1	1	30	1	3.70E-11	3.70E-11	.42
3	1	1	26	1	2.81E-11	2.81E-11	.62
3	2	1	56	1	3.00E-11	3.00E-11	1.00
4	1	1	15	1	2.17E-11	2.17E-11	.69
4	2	2	15	2	2.01E-11	1.81E-11	.69
4	3	3	24	3	2.53E-11	2.09E-11	.92
5	1	1	13	1	2.20E-11	2.20E-11	1.02
5	2	2	13	1	2.03E-11	1.99E-11	1.03
5	3	3	13	1	2.24E-11	1.60E-11	1.01
5	4	4	33	1	2.37E-11	6.67E-12	1.44
6	1	1	13	1	2.09E-11	2.09E-11	1.36
6	2	2	13	2	2.16E-11	2.04E-11	1.36
6	3	3	13	2	2.89E-11	2.03E-11	1.33
6	4	4	21	4	2.02E-10	3.63E-11	1.76
6	5	4	31	1	2.64E-11	1.09E-11	2.10

FINAL EIGENVALUES

2.393019487013E+00 -1.245936323012E+00 1.188248083996E+00 -2.273759446561E-01 -9.705562962258E-02

Fig. 1

REFERENCES

1. ARNURIUS, D.E. Private communication, 1973.
2. BAUER, F.L. Das Verfaheren der Treppeniteration und Verwandte Verfaheren zur Lösung algebraischer Eigenvertprobleme. *Z. Angew. Math. Phys. 8* (1957), 214-235.
3. GARBOW, B.S. Private communication, 1973, 1975, 1976.
4. MCCORMICK, S.F., AND NOE, T. Simultaneous iteration for the matrix eigenvalue problem. *J. Lin. Alg. Appl. 16* (1977), 43-56.
5. NIKOLAI, P.J., AND TSAO, N. The ARL Linear Algebra Library Handbook, ARL TR 74-0106, Aerospace Research Labs., Air Force Systems Command, U.S. Air Force, Wright-Patterson AFB, Ohio, distributed by National Technical Information Services Clearinghouse, Arlington, Va., 1974.
6. PETERS, G., AND WILKINSON, J.H., $Ax = \lambda Bx$ and the generalized eigenproblem. *SIAM J. Numer. Anal. 7* (1970), 479-492.
7. RUTISHAUSER, H. Computational aspects of F.L. Bauer's simultaneous iteration method. *Numer. Math. 13* (1969), 4-13.
8. RUTISHAUSER, H. Simultaneous iteration method for symmetric matrices. *Numer. Math. 16* (1970), 205-223.
9. SMITH, B.T., et al. Matrix eigensystem routines—EISPACK guide. In *Lecture Notes in Computer Science 6*, Springer-Verlag, New York, 1974.
10. WILKINSON, J.H., AND REINSCH, C. *Linear Algebra, Handbook for Automatic Computation II*, Springer-Verlag, New York, 1971.

ALGORITHM

```
      SUBROUTINE SIMITZ(N, P, KM, EPS, IP, OP, INF, EM, X, MN, D, WK)
C*********************************************************************
C     IDENTIFICATION
C       SIMITZ - ITERATIVE COMPUTATION OF EIGENVALUES LARGEST IN MAGNI-
C                TUDE AND CORRESPONDING EIGENVECTORS OF A REAL GENERAL-
C                IZED SYMMETRIC MATRIX
C       FORTRAN SUBROUTINE SUBPROGRAM
C       US AIR FORCE FLIGHT DYNAMICS LABORATORY
C       WRIGHT-PATTERSON AFB, OHIO  45433
C     PURPOSE
C       A REAL N-SQUARE MATRIX C IS B-SYMMETRIC RELATIVE TO AN N-SQUARE
C       POSITIVE DEFINITE MATRIX B IN CASE BC = (C-TRANSPOSED)B.
C       GIVEN AS OPTIONAL INPUT A SET OF P INITIAL APPROXIMATE
C       EIGENVECTORS OF A REAL N-SQUARE B-SYMMETRIC MATRIX C CORRES-
C       PONDING TO P EIGENVALUES OF C LARGEST IN MAGNITUDE, SIMITZ COM-
C       PUTES EM EIGENVALUES AND EM CORRESPONDING EIGENVECTORS TO A
C       PRECISION DEPENDENT ON THE STRUCTURE OF B AND C AND ON A GIVEN
C       TOLERANCE EPS.  THE MATRIX B IS PRESENTED TO SIMITZ AS AN ALGO-
C       RITHM FOR CALCULATING THE STANDARD INNER PRODUCT (W, BZ) =
C       (W-TRANSPOSED)BZ GIVEN COLUMN N-VECTORS W AND Z IMPLEMENTED AS
C       A FORTRAN COMPATIBLE REAL FUNCTION SUBPROGRAM.  THE MATRIX C IS
C       PRESENTED AS A SUBROUTINE SUBPROGRAM WHICH GIVEN A COLUMN
C       N-VECTOR Z CALCULATES ITS IMAGE W = CZ UNDER THE MATRIX C.
C       DEPENDING ON THE CHOICE OF B AND C, SIMITZ APPLIES TO A WIDE
C       VARIETY OF SYMMETRIC EIGENPROBLEMS.
C     CONTROL
C
C       DIMENSION X(MN,P), D(P), WK(K)
C       INTEGER P, EM
C       REAL IP
C       EXTERNAL IP, INF, OP
C       .
C       .
C       .
C       CALL SIMITZ(N, P, KM, EPS, IP, OP, INF, EM, X, MN, D, WK)
C
C       WHERE
C         N   IS AN INTEGER INPUT VARIABLE, THE ORDER OF THE MATRIX C.
C         P   IS AN INTEGER INPUT VARIABLE, THE NUMBER OF SIMULTANEOUS
C             ITERATION VECTORS.
C         KM  AS AN INTEGER INPUT VARIABLE IS IN MAGNITUDE THE MAXIMUM
C             NUMBER OF ITERATION STEPS TO BE EXECUTED.  IF KM IDENTIFIES
C             A NEGATIVE VALUE THEN P INITIAL APPROXIMATE EIGENVECTORS
C             ARE ASSUMED TO BE PRESENT IN THE ARRAY X.  OTHERWISE SIMITZ
C             SUPPLIES RANDOM INITIAL EIGENVECTORS.
C         KM  AS AN INTEGER OUTPUT VARIABLE IDENTIFIES THE NUMBER KS OF
C             ITERATION STEPS FINALLY USED IN THE CALCULATION OF EM
C             EIGENVECTORS.
C         EPS IS A REAL INPUT VARIABLE, THE TOLERANCE FOR ACCEPTING
C             EIGENVECTORS.  AS SOON AS SUCCESSIVE ITERATES OF THE RITZ
```

```
C            VALUES ABS(D(H+1)) DIFFER BY LESS THAN ABS(D(H+1))*EPS/10.0      510
C            THEN D(H+1) IS ACCEPTED AS AN EIGENVALUE AND H, THE NUMBER       520
C            OF PREVIOUSLY ACCEPTED EIGENVALUES, IS INCREASED BY 1.  AS       530
C            SOON AS THE ERROR QUANTITIES F(I), NORMALIZED RESIDUALS,         540
C            SATISFY D(I)*F(I)/(D(I) - D(P)) .LT. EPS, THEN G, THE NUM-       550
C            BER OF ALREADY ACCEPTED RITZ VECTORS, IS INCREASED TO            560
C            G + L, I = G + 1, ..., L.  THE F(I) ARE DISCOUNTED WITH          570
C            SUCCESSIVE ITERATIONS TO FORCE CONVERGENCE IN CASE OF UN-        580
C            FORTUNATE CHOICE OF PARAMETERS.  IF M SIGNIFICANT DIGITS         590
C            OF ACCURACY ARE REQUIRED OF THE EIGENVALUES, THEN SET            600
C            EPS EQUAL TO 10.0**(-M) AS A GENERAL RULE.                       610
C         IP IS AN EXTERNAL INPUT VARIABLE, THE NAME OF A FORTRAN COM-        620
C            PATIBLE REAL FUNCTION SUBPROGRAM OF THE FORM IP(N, Z, W)         630
C            WHICH MUST RETURN THE INNER PRODUCT (W, BZ) =                    640
C            (W-TRANSPOSED)BZ OF THE VECTORS IDENTIFIED BY THE N-ARRAYS       650
C            Z AND W RELATIVE TO THE POSITIVE DEFINITE MATRIX B.              660
C         OP IS AN EXTERNAL INPUT VARIABLE, THE NAME OF A FORTRAN COM-        670
C            PATIBLE SUBROUTINE SUBPROGRAM OF THE FORM OP(N, Z, W)            680
C            WHICH MUST CALCULATE THE IMAGE W OF THE VECTOR IDENTIFIED        690
C            BY THE N-ARRAY Z UNDER THE N-SQUARE MATRIX C WITHOUT OVER-       700
C            WRITING Z.                                                       710
C        INF IS AN EXTERNAL INPUT VARIABLE, THE NAME OF A FORTRAN COM-        720
C            PATIBLE SUBROUTINE SUBPROGRAM WHICH MAY BE USED FOR              730
C            OBTAINING INFORMATION OR TO EXERT CONTROL DURING EXECUTION       740
C            OF SIMITZ.  INF HAS THE FORM INF(KS, G, H, F) WHERE              750
C              KS IS AN INTEGER OUTPUT VARIABLE, THE NUMBER OF THE NEXT       760
C                 ITERATION STEP.                                             770
C              G  IS AN INTEGER OUTPUT VARIABLE, THE NUMBER OF ALREADY        780
C                 ACCEPTED EIGENVECTORS.                                      790
C              H  IS AN INTEGER OUTPUT VARIABLE, THE NUMBER OF ALREADY        800
C                 ACCEPTED EIGENVALUES.                                       810
C              F  IS A REAL OUTPUT VARIABLE P-ARRAY, ERROR QUANTITIES         820
C                 MEASURING RESPECTIVELY THE STATE OF CONVERGENCE OF          830
C                 THE P SIMULTANEOUS ITERATION VECTORS.  IN ADDITION,         840
C                 IF CONVERGENCE FAILS IN SUBROUTINE IMTQL2 AFTER G           850
C                 EIGENVECTORS HAVE BEEN ACCEPTED, THEN F(G+1) IS RE-         860
C                 PLACED BY 1000.*FLOAT(IERR) WHERE IERR IS THE ERROR         870
C                 INDICATOR OUTPUT BY IMTQL2.  EACH ELEMENT OF THE ARRAY      880
C                 F IS INITIALLY SET BY SIMITZ TO THE VALUE 4.0.              890
C        EM  AS AN INTEGER INPUT VARIABLE IS THE NUMBER OF EIGENVALUES        900
C            TO BE COMPUTED, 0 .LT. EM .LT. P .LE. N .LE. MN.                 910
C        EM  AS AN INTEGER OUTPUT VARIABLE IS THE NUMBER OF EIGENVECTORS      920
C            COMPUTED THROUGH KM ITERATION STEPS.                             930
C        X   AS A REAL N-BY-P INPUT ARRAY IS A SET OF P OPTIONAL INITIAL      940
C            APPROXIMATE EIGENVECTORS X(I,1), ..., X(I,P), I = 1, ...,        950
C            N, INTERPRETED BY SIMITZ IF KM IS NEGATIVE.                      960
C        X   AS A REAL N-BY-P OUTPUT ARRAY IS A SET OF EM EIGENVECTORS        970
C            X(I,1), ..., X(I,EM), I = 1, ..., N, COMPUTED THROUGH            980
C            IABS(KM) ITERATION STEPS WITH THE REMAINDER OF X CONSISTING      990
C            OF P - EM APPROXIMATE EIGENVECTORS.  THE N-BY-P MATRIX X        1000
C            SATISFIES (X-TRANSPOSED)BX = I, THAT IS, THE EIGENVECTORS       1010
C            OF C ARE B-ORTHONORMAL.                                         1020
C        MN  IS AN INTEGER INPUT VARIABLE WHICH IDENTIFIES THE LEADING       1030
C            DIMENSION IN THE CALLING PROGRAM OF THE ARRAY X.                1040
C        D   IS A REAL OUTPUT P-ARRAY OF WHICH D(1), ..., D(EM) ARE THE      1050
C            EIGENVALUES OF C LARGEST IN MAGNITUDE IN DECREASING ORDER       1060
C            CORRESPONDING TO THE COMPUTED EIGENVECTORS X(I,1), ...,         1070
C            X(I,EM), I = 1, ..., N.  D(EM+1), ..., D(P-1) CONTAIN           1080
C            APPROXIMATIONS TO PROGRESSIVELY SMALLER SUCH EIGENVALUES.       1090
C            D(P) CONTAINS THE MOST RECENTLY COMPUTED VALUE OF E, WHERE       1100
C            THE INTERVAL (-E, E) IS THE INTERVAL OVER WHICH THE             1110
C            CHEBYSHEV ACCELERATION WAS PERFORMED.                           1120
C        WK  THE INITIAL LOCATION OF AT LEAST P**2 + 3*P + 2*N = K           1130
C            CONSECUTIVE STORAGE LOCATIONS WHICH MAY NOT BE OVER-            1140
C            WRITTEN WHILE SIMITZ IS IN EXECUTION.                           1150
C      OTHER PROGRAMMING INFORMATION                                         1160
C        SIMITZ EMPLOYS A DATA STATEMENT TO ASSIGN TO A MACHINE DEPEN-       1170
C        DENT REAL VARIABLE MT THE QUOTIENT OF THE SMALLEST POSITIVE         1180
C        REAL VALUE REPRESENTABLE BY FORTRAN AND THE SMALLEST REAL VALUE     1190
C        WHOSE SUM WITH 1.0 EXCEEDS 1.0.                                     1200
C                                                                            1210
C        THE PERFORMANCE OF SIMITZ IS STRONGLY DEPENDENT UPON THE CHOICE     1220
C        OF INPUT PARAMETERS AND UPON THE CAREFUL PREPARATION OF THE         1230
C        SUBPROGRAMS IP AND OP.  THE USER SHOULD CONSIDER USING HIS OWN      1240
C        ACTIVE SUBROUTINE INF TO MONITOR PROGRESS OF SIMITZ RELATIVE TO     1250
C        HIS CHOICE OF INPUT PARAMETERS IF NO INFORMATION IS OTHERWISE       1260
C        AVAILABLE CONCERNING THE LOCATIONS OF THE RELEVANT EIGENVALUES.     1270
C        RECALL THAT SIMITZ MAY BE REENTERED WITH KM .LT. 0 WITHOUT LOSS     1280
```

```
C        OF INFORMATION TO PERMIT CONSERVATIVE INITIAL CHOICES OF          1290
C        ABS(KM), EPS AND P.                                               1300
C      OTHER PROGRAMS REQUIRED                                             1310
C        FUNCTION RANF                                                     1320
C            RETURNS UNIFORMLY DISTRIBUTED RANDOM NUMBERS ON THE OPEN      1330
C            INTERVAL (0, 1) GIVEN ANY ONE ARGUMENT OF ANY TYPE.           1340
C        SUBROUTINE TRED2                                                  1350
C            IS THE EISPACK (4) PROGRAM WHICH COMPUTES A HOUSEHOLDER       1360
C            TRIDIAGONAL FORM OF A REAL SYMMETRIC MATRIX.                  1370
C        SUBROUTINE IMTQL2                                                 1380
C            IS THE EISPACK PROGRAM WHICH COMPUTES THE EIGENVALUES AND     1390
C            ORTHONORMAL EIGENVECTORS OF A SYMMETRIC TRIDIAGONAL MATRIX.   1400
C        FUNCTION IP                                                       1410
C            IS DESCRIBED ABOVE.                                           1420
C        SUBROUTINE OP                                                     1430
C            IS DESCRIBED ABOVE.                                           1440
C        SUBROUTINE INF                                                    1450
C            IS DESCRIBED ABOVE.                                           1460
C      METHOD                                                              1470
C        SIMITZ REPRESENTS RESULTS OF EXTENSIVE MODIFICATIONS AND TESTS    1480
C        OF SUBROUTINE RITZIT (1), AN ANSI FORTRAN TRANSLATION OF THE      1490
C        ALGOL 60 PROCEDURE OF THE SAME NAME (3).  THE BASIC RUTISHAUSER   1500
C        -REINSCH ALGORITHM IS PRESERVED.                                  1510
C      REFERENCES                                                          1520
C        (1) PAUL J. NIKOLAI AND NAI-KUAN TSAO, THE ARL LINEAR ALGEBRA     1530
C            LIBRARY HANDBOOK, ARL TR 74-0106, AEROSPACE RESEARCH LABOR-   1540
C            ATORIES, WRIGHT-PATTERSON AFB, OHIO, 1974.                    1550
C        (2) HEINZ RUTISHAUSER, COMPUTATIONAL ASPECTS OF F.L. BAUER S      1560
C            SIMULTANEOUS ITERATION METHOD, NUMER. MATH. 13(1969), 4-13.   1570
C        (3) ----------------, SIMULTANEOUS ITERATION METHOD FOR SYM-      1580
C            METRIC MATRICES, NUMER. MATH. 16(1970), 205-223.              1590
C        (4) B.T. SMITH ET AL, MATRIX EIGENSYSTEM ROUTINES-EISPACK         1600
C            GUIDE, LECTURE NOTES IN COMPUTER SCIENCE 6, SPRINGER-VERLAG   1610
C            NEW YORK, 1974.                                               1620
C                                                                          163Ø
C        PLEASE REFER QUESTIONS, COMMENTS OR SUGGESTIONS TO                164Ø
C                                                                          165Ø
C        PAUL J. NIKOLAI                                                   166Ø
C        AFFDL/FBR                                                         167Ø
C        WRIGHT-PATTERSON AFB, OH 45433                                    168Ø
C        (513)-255-535Ø                                                    169Ø
C                                                                          17ØØ
C*************************************************************************  171Ø
      EXTERNAL    INF,     IP,     OP                                      172Ø
      INTEGER      EM,      G,      H,      I,     IG,     IK,      J,     173Ø
     *             JK,     JP,      K,     KM,     KS,      L,     LF,     174Ø
     *             L1,      M,     MN,     M1,      N,      P,     Z1,     175Ø
     *             Z2                                                      176Ø
      LOGICAL    ORIG                                                     177Ø
      REAL          D,      E,    EPS,     E1,     E2,     IP,     MT,     178Ø
     *              S,      T,     WK,      X                             179Ø
      DIMENSION X(MN,1), D(1), WK(P,P,1)                                  18ØØ
C                                                                         181Ø
      DATA MT /.22Ø36Ø641585Ø62E-279/                                     182Ø
C                                                                         183Ø
C        THE LOCAL VARIABLE ARRAYS FROM (3) ARE ASSIGNED TO THE           184Ø
C                VARIABLE ARRAY WK AS FOLLOWS                             185Ø
C                                                                         186Ø
C        WK(I,J,1) = B(I,J), I, J = 1, ..., P.                           187Ø
C        WK(I,1,2) = CX(I), I = 1, ..., P.                                188Ø
C        WK(I,2,2) = F(I), I = 1, ..., P.                                 189Ø
C        WK(I,3,2) = RQ(I), I = 1, ..., P.                                19ØØ
C        WK(I,4,2) = U(I), I = 1, ..., N.                                 191Ø
C        WK(I+N,4,2) = W(I), I = 1, ..., N.                               192Ø
C        NOT NEEDED ARE V(I), I = 1, ..., N, R(I), I = 1, ..., P, AND     193Ø
C        Q(I,J), I, J = 1, ..., P.                                        194Ø
C                                                                         195Ø
C              THE NEXT STATEMENT IS START.                               196Ø
      E = .ØE+ØØ                                                          197Ø
      G = Ø                                                               198Ø
      IG = 1                                                              199Ø
      H = Ø                                                               2ØØØ
      Z1 = Ø                                                              2Ø1Ø
      Z2 = Ø                                                              2Ø2Ø
      KS = Ø                                                              2Ø3Ø
      M = 1                                                               2Ø4Ø
      WK(P,1,2) = .ØE+ØØ                                                  2Ø5Ø
```

```
      DO 1Ø L = 1, P                                                    2Ø6Ø
        WK(L,2,2) = .4E+Ø1                                              2Ø7Ø
        WK(L,3,2) = .ØE+ØØ                                              2Ø8Ø
 1Ø   CONTINUE                                                          2Ø9Ø
      IF (KM) 5Ø, 5Ø, 2Ø                                                21ØØ
 2Ø   DO 4Ø L = 1, P                                                    211Ø
        DO 3Ø J = 1, N                                                  212Ø
          X(J,L) = .2E+Ø1*RANF(.2E+Ø1) - .1E+Ø1                         213Ø
 3Ø     CONTINUE                                                       214Ø
 4Ø   CONTINUE                                                          215Ø
 5Ø   KM = IABS(KM)                                                     216Ø
      ASSIGN 6Ø TO IK                                                   217Ø
      LF = IG                                                           218Ø
      L1 = P                                                            219Ø
      GO TO 99Ø                                                         22ØØ
C        RAYLEIGH-RITZ STEP                                             221Ø
C              STATEMENT 6Ø IS LOOP.                                    222Ø
 6Ø   DO 8Ø K = IG, P                                                   223Ø
        CALL OP(N, X(1,K), WK(1,4,2))                                   224Ø
        DO 7Ø J = 1, N                                                  225Ø
          X(J,K) = WK(J,4,2)                                            226Ø
 7Ø     CONTINUE                                                       227Ø
 8Ø   CONTINUE                                                          228Ø
      ASSIGN 9Ø TO IK                                                   229Ø
      LF = IG                                                           23ØØ
      L1 = P                                                            231Ø
      GO TO 99Ø                                                         232Ø
 9Ø   IF (KS) 15Ø, 1ØØ, 15Ø                                            233Ø
C        MEASURES AGAINST UNHAPPY CHOICE OF INITIAL VECTORS             234Ø
 1ØØ  DO 13Ø K = 1, P                                                   235Ø
        IF (WK(K,K,1)) 13Ø, 11Ø, 13Ø                                    236Ø
 11Ø    DO 12Ø I = 1, N                                                 237Ø
          X(I,K) = .2E+Ø1*RANF(.2E+Ø1) - .1E+Ø1                         238Ø
 12Ø    CONTINUE                                                        239Ø
        KS = 1                                                          24ØØ
 13Ø  CONTINUE                                                          241Ø
      IF (KS - 1) 15Ø, 14Ø, 15Ø                                         242Ø
 14Ø  ASSIGN 6Ø TO IK                                                   243Ø
      LF = 1                                                            244Ø
      L1 = P                                                            245Ø
      GO TO 99Ø                                                         246Ø
 15Ø  DO 18Ø K = IG, P                                                  247Ø
        DO 17Ø L = K, P                                                 248Ø
          S = .ØE+ØØ                                                    249Ø
          DO 16Ø I = L, P                                               25ØØ
            S = S + WK(I,K,1)*WK(I,L,1)                                 251Ø
 16Ø      CONTINUE                                                      252Ø
          WK(L,K,1) = -S                                                253Ø
 17Ø    CONTINUE                                                       254Ø
 18Ø  CONTINUE                                                          255Ø
      CALL TRED2(P, P - G, WK(IG,IG,1), D(IG), WK(1,4,2), WK(IG,IG,1))  256Ø
      CALL IMTQL2(P, P - G, D(IG), WK(1,4,2), WK(IG,IG,1), L)           257Ø
      WK(IG,2,2) = AMAX1(WK(IG,2,2), .1E+Ø4*FLOAT(L))                   258Ø
      DO 19Ø K = IG, P                                                  259Ø
        D(K) = SQRT(AMAX1(-D(K), .ØE+ØØ))                               26ØØ
 19Ø  CONTINUE                                                          261Ø
C        REORDERING EIGENVALUES AND EIGENVECTORS ACCORDING TO SIZE OF   262Ø
C        THE FORMER IS ACCOMPLISHED IN SUBROUTINE IMTQL2.               263Ø
      DO 23Ø J = 1, N                                                   264Ø
        DO 21Ø K = IG, P                                                265Ø
          S = .ØE+ØØ                                                    266Ø
          DO 2ØØ L = IG, P                                              267Ø
            S = S + X(J,L)*WK(L,K,1)                                    268Ø
 2ØØ      CONTINUE                                                      269Ø
          WK(K,4,2) = S                                                 27ØØ
 21Ø    CONTINUE                                                       271Ø
        DO 22Ø K = IG, P                                                272Ø
          X(J,K) = WK(K,4,2)                                            273Ø
 22Ø    CONTINUE                                                       274Ø
 23Ø  CONTINUE                                                          275Ø
      KS = KS + 1                                                       276Ø
      E = AMAX1(D(P), E)                                                277Ø
C        RANDOMIZATION                                                  278Ø
      IF (3 - Z1) 26Ø, 24Ø, 24Ø                                         279Ø
 24Ø  DO 25Ø J = 1, N                                                   28ØØ
        X(J,P) = .2E+Ø1*RANF(.2E+Ø1) - .1E+Ø1                           281Ø
```

```
 25Ø  CONTINUE                                                          282Ø
      JP = P - 1                                                        283Ø
      ASSIGN 26Ø TO IK                                                  284Ø
      LF = P                                                            285Ø
      L1 = P                                                            286Ø
      GO TO 99Ø                                                         287Ø
C        COMPUTE CONTROL QUANTITIES CX(I).                              288Ø
 26Ø  DO 31Ø K = IG, JP                                                 289Ø
        S = (D(K) - E)*(D(K) + E)                                       29ØØ
        IF (S) 27Ø, 27Ø, 28Ø                                            291Ø
 27Ø    WK(K,1,2) = .ØE+ØØ                                             292Ø
        GO TO 31Ø                                                       293Ø
 28Ø    IF (E) 3ØØ, 29Ø, 3ØØ                                           294Ø
 29Ø    WK(K,1,2) = .1E+Ø4 + ALOG(D(K))                                295Ø
        GO TO 31Ø                                                       296Ø
 3ØØ    WK(K,1,2) = ALOG((D(K) + SQRT(S))/E)                           297Ø
 31Ø  CONTINUE                                                          298Ø
C        ACCEPTANCE TEST FOR EIGENVALUES INCLUDING ADJUSTMENT OF EM AND  299Ø
C        H SUCH THAT D(EM) .GT. E, D(H) .GT. E AND D(EM) DOES NOT       3ØØØ
C        OSCILLATE STRONGLY                                             3Ø1Ø
      I = Z1 - 1                                                        3Ø2Ø
      K = G                                                             3Ø3Ø
 32Ø  K = K + 1                                                         3Ø4Ø
        IF (EM - K) 37Ø, 33Ø, 33Ø                                       3Ø5Ø
 33Ø    IF (D(K) - E) 36Ø, 36Ø, 34Ø                                    3Ø6Ø
 34Ø    IF (I) 32Ø, 32Ø, 35Ø                                           3Ø7Ø
 35Ø    IF (D(K) - .999E+ØØ*WK(K,3,2)) 36Ø, 36Ø, 32Ø                   3Ø8Ø
 36Ø  CONTINUE                                                          3Ø9Ø
      EM = K - 1                                                        31ØØ
C               STATEMENT 37Ø IS EX4.                                   311Ø
 37Ø  IF (EM) 38Ø, 113Ø, 38Ø                                            312Ø
 38Ø  K = H                                                             313Ø
      S = .1E+Ø1 + .1E+ØØ*EPS                                           314Ø
 39Ø  K = K + 1                                                         315Ø
        IF (D(K)) 4ØØ, 41Ø, 4ØØ                                         316Ø
 4ØØ    IF (D(K) - S*WK(K,3,2)) 39Ø, 39Ø, 41Ø                          317Ø
 41Ø  CONTINUE                                                          318Ø
      H = K - 1                                                         319Ø
      K = EM                                                            32ØØ
 42Ø  K = K + 1                                                         321Ø
        IF (K - H) 43Ø, 43Ø, 45Ø                                        322Ø
 43Ø    IF (D(K) - E) 44Ø, 44Ø, 42Ø                                    323Ø
 44Ø  CONTINUE                                                          324Ø
      H = K - 1                                                         325Ø
C       ACCEPTANCE TEST FOR EIGENVECTORS                                326Ø
 45Ø  L = G                                                             327Ø
      E2 = .ØE+ØØ                                                       328Ø
      DO 59Ø K = IG, JP                                                 329Ø
        IF (K - (L + 1)) 51Ø, 46Ø, 51Ø                                  33ØØ
C        CHECK FOR NESTED EIGENVALUES                                   331Ø
 46Ø    L = K                                                          332Ø
        L1 = K                                                          333Ø
        S = .5E+ØØ/FLOAT(KS)                                            334Ø
        T = .1E+Ø1/FLOAT(KS*M)                                          335Ø
 47Ø    L = L + 1                                                      336Ø
          IF (L - JP) 48Ø, 48Ø, 49Ø                                     337Ø
 48Ø      IF (WK(L,1,2)*(WK(L,1,2) + S) + T - WK(L-1,1,2)*WK(L-1,1,2))  338Ø
     *        49Ø, 49Ø, 47Ø                                            339Ø
 49Ø    CONTINUE                                                       34ØØ
        L = L - 1                                                       341Ø
C               THE NEXT STATEMENT IS EX5.                              342Ø
        IF (L - H) 51Ø, 51Ø, 5ØØ                                        343Ø
 5ØØ    L = L1 - 1                                                     344Ø
        GO TO 6ØØ                                                       345Ø
 51Ø    CALL OP(N, X(1,K), WK(1,4,2))                                  346Ø
        S = .ØE+ØØ                                                      347Ø
        DO 54Ø J = 1, L                                                 348Ø
          IF (ABS(D(J) - D(K)) - .1E-Ø1*D(K)) 52Ø, 54Ø, 54Ø             349Ø
 52Ø      T = IP(N, WK(1,4,2), X(1,J))                                  35ØØ
          DO 53Ø I = 1, N                                               351Ø
            WK(I,4,2) = WK(I,4,2) - T*X(I,J)                            352Ø
 53Ø      CONTINUE                                                      353Ø
          S = S + T*T                                                   354Ø
 54Ø    CONTINUE                                                       355Ø
        T = .1E+Ø1                                                      356Ø
        IF (S .NE. .ØE+ØØ) T = IP(N, WK(1,4,2), WK(1,4,2))              357Ø
```

```
        E2 = AMAX1(E2, SQRT(T/(S + T)))                                    358Ø
        IF (K - L) 59Ø, 55Ø, 59Ø                                           359Ø
C        TEST FOR ACCEPTANCE OF GROUP OF EIGENVECTORS                      36ØØ
 55Ø    IF (L .GE. EM .AND. D(EM)*WK(EM,2,2) .LT. EPS*(D(EM) - E))        361Ø
     *     G = EM                                                          362Ø
        IF (E2 - WK(L,2,2)) 56Ø, 58Ø, 58Ø                                  363Ø
 56Ø    DO 57Ø J = L1, L                                                   364Ø
          WK(J,2,2) = E2                                                   365Ø
 57Ø    CONTINUE                                                           366Ø
 58Ø    IF (L .LE. EM .AND. D(L)*WK(L,2,2) .LT. EPS*(D(L) - E)) G = L      367Ø
 59Ø  CONTINUE                                                            368Ø
C        ADJUST M.                                                         369Ø
C              STATEMENT 6ØØ IS EX6.                                       37ØØ
 6ØØ  IG = G + 1                                                          371Ø
      IF (E - .4E-Ø1*D(1)) 61Ø, 61Ø, 62Ø                                  372Ø
 61Ø  M = 1                                                               373Ø
      K = 1                                                               374Ø
      GO TO 63Ø                                                           375Ø
 62Ø  E2 = .2E+Ø1/E                                                       376Ø
      E1 = .51E+ØØ*E2                                                     377Ø
      K = 2*INT(.4E+Ø1/AMIN1(WK(1,1,2), .4E+Ø1))                          378Ø
      M = MINØ(M, K)                                                      379Ø
C        REDUCE EM IF CONVERGENCE WOULD BE TOO SLOW.                       38ØØ
 63Ø  IF (WK(EM,2,2)) 64Ø, 69Ø, 64Ø                                       381Ø
 64Ø  IF (FLOAT(KS) - .9E+ØØ*FLOAT(KM)) 65Ø, 69Ø, 69Ø                     382Ø
 65Ø  S = FLOAT(K)*WK(EM,1,2)                                             383Ø
      IF (S - .5E-Ø1) 66Ø, 67Ø, 67Ø                                       384Ø
 66Ø  T = .5E+ØØ*S*WK(EM,1,2)                                             385Ø
      GO TO 68Ø                                                           386Ø
 67Ø  T = WK(EM,1,2) + ALOG(.5E+ØØ + .5E+ØØ*EXP(-.2E+Ø1*S))/FLOAT(K)      387Ø
 68Ø  S = ALOG(D(EM)*WK(EM,2,2)/(EPS*(D(EM) - E)))                        388Ø
      IF (S*FLOAT(KS) .GT. T*FLOAT((KM - KS)*KM)) EM = EM - 1             389Ø
C              STATEMENT 69Ø IS EX2.                                       39ØØ
 69Ø  DO 7ØØ K = IG, JP                                                   391Ø
        WK(K,3,2) = D(K)                                                  392Ø
 7ØØ  CONTINUE                                                            393Ø
      CALL INF(KS, G, H, WK(1,2,2))                                       394Ø
      IF (G .GE. EM .OR. KS .GE. KM) GO TO 113Ø                           395Ø
C              STATEMENT 71Ø IS EX1.                                       396Ø
 71Ø  IF (KS + M - KM) 73Ø, 73Ø, 72Ø                                      397Ø
 72Ø  Z2 = -1                                                             398Ø
      IF (M .GT. 1) M = 2*((KM - KS + 1)/2)                               399Ø
 73Ø  M1 = M                                                              4ØØØ
C        SHORTCUT LAST INTERMEDIATE BLOCK IF ALL F(I) ARE SUFFICIENTLY     4Ø1Ø
C        SMALL.                                                            4Ø2Ø
      IF (L - EM) 78Ø, 74Ø, 74Ø                                           4Ø3Ø
 74Ø  S = D(EM)*WK(EM,2,2)/(EPS*(D(EM) - E))                              4Ø4Ø
      T = S*S - .1E+Ø1                                                    4Ø5Ø
      IF (T) 6Ø, 6Ø, 75Ø                                                  4Ø6Ø
 75Ø  S = ALOG(S + SQRT(T))/(WK(EM,1,2) - WK(H+1,1,2))                    4Ø7Ø
      M1 = 2*INT(.5E+ØØ*S + .1Ø1E+Ø1)                                     4Ø8Ø
      IF (M1 - M) 77Ø, 77Ø, 76Ø                                           4Ø9Ø
 76Ø  M1 = M                                                              41ØØ
      GO TO 78Ø                                                           411Ø
 77Ø  Z2 = -1                                                             412Ø
C        CHEBYSHEV ITERATION                                               413Ø
 78Ø  IF (M - 1) 9ØØ, 79Ø, 82Ø                                            414Ø
 79Ø  DO 81Ø K = IG, P                                                    415Ø
        CALL OP(N, X(1,K), WK(1,4,2))                                     416Ø
        DO 8ØØ I = 1, N                                                   417Ø
          X(I,K) = WK(I,4,2)                                              418Ø
 8ØØ    CONTINUE                                                          419Ø
 81Ø  CONTINUE                                                            42ØØ
      GO TO 9ØØ                                                           421Ø
 82Ø  L1 = M1 - 4                                                         422Ø
      DO 89Ø K = IG, P                                                    423Ø
        CALL OP(N, X(1,K), WK(1,4,2))                                     424Ø
        DO 83Ø I = 1, N                                                   425Ø
          IK = I + N                                                      426Ø
          WK(IK,4,2) = E1*WK(I,4,2)                                       427Ø
 83Ø    CONTINUE                                                          428Ø
        CALL OP(N, WK(N+1,4,2), WK(1,4,2))                                429Ø
        DO 84Ø I = 1, N                                                   43ØØ
          X(I,K) = E2*WK(I,4,2) - X(I,K)                                  431Ø
 84Ø    CONTINUE                                                          432Ø
        IF (L1) 89Ø, 85Ø, 85Ø                                             433Ø
 85Ø    DO 88Ø J = 4, M1, 2                                               434Ø
```

```
          CALL OP(N, X(1,K), WK(1,4,2))                                  435Ø
          DO 86Ø I = 1, N                                                436Ø
            IK = I + N                                                   437Ø
            WK(IK,4,2) = E2*WK(I,4,2) - WK(IK,4,2)                       438Ø
 86Ø      CONTINUE                                                       439Ø
          CALL OP(N, WK(N+1,4,2), WK(1,4,2))                             44ØØ
          DO 87Ø I = 1, N                                                441Ø
            X(I,K) = E2*WK(I,4,2) - X(I,K)                               442Ø
 87Ø      CONTINUE                                                       443Ø
 88Ø    CONTINUE                                                         444Ø
 89Ø  CONTINUE                                                           445Ø
 9ØØ  ASSIGN 91Ø TO IK                                                   446Ø
      LF = IG                                                            447Ø
      L1 = P                                                             448Ø
      GO TO 99Ø                                                          449Ø
C        DISCOUNTING THE ERROR QUANTITIES F                              45ØØ
 91Ø  IF (G - H) 92Ø, 97Ø, 97Ø                                           451Ø
 92Ø  IF (M - 1) 95Ø, 93Ø, 95Ø                                           452Ø
 93Ø  DO 94Ø K = IG, H                                                   453Ø
        WK(K,2,2) = WK(K,2,2)*(D(H+1)/D(K))                              454Ø
 94Ø  CONTINUE                                                           455Ø
      GO TO 97Ø                                                          456Ø
 95Ø  T = EXP(-FLOAT(M1)*WK(H+1,1,2))                                    457Ø
      DO 96Ø K = IG, H                                                   458Ø
        S = EXP(-FLOAT(M1)*(WK(K,1,2) - WK(H+1,1,2)))                    459Ø
        WK(K,2,2) = S*WK(K,2,2)*(.1E+Ø1 + T*T)/(.1E+Ø1 + (S*T)*(S*T))    46ØØ
 96Ø  CONTINUE                                                           461Ø
 97Ø  KS = KS + M1                                                       462Ø
      Z2 = Z2 - M1                                                       463Ø
C        POSSIBLE REPETITION OF INTERMEDIATE STEPS                       464Ø
      IF (Z2) 98Ø, 71Ø, 71Ø                                              465Ø
 98Ø  Z1 = Z1 + 1                                                        466Ø
      Z2 = 2*Z1                                                          467Ø
      M = M + M                                                          468Ø
      GO TO 6Ø                                                           469Ø
C        PERFORMS ORTHONORMALIZATION OF COLUMNS 1 THROUGH L1 OF ARRAY    47ØØ
C        X ASSUMING THAT COLUMNS 1 THROUGH LF - 1 ARE ALREADY ORTHO-     471Ø
C        NORMAL                                                          472Ø
 99Ø  DO 112Ø K = LF, L1                                                 473Ø
        ORIG = .TRUE.                                                    474Ø
 1ØØØ   T = .ØE+ØØ                                                       475Ø
        JK = K - 1                                                       476Ø
        IF (JK) 1Ø4Ø, 1Ø4Ø, 1Ø1Ø                                         477Ø
 1Ø1Ø   DO 1Ø3Ø I = 1, JK                                                478Ø
          S = IP(N, X(1,I), X(1,K))                                      479Ø
          IF (ORIG) WK(K,I,1) = S                                        48ØØ
          T = T + S*S                                                    481Ø
          DO 1Ø2Ø J = 1, N                                               482Ø
            X(J,K) = X(J,K) - S*X(J,I)                                   483Ø
 1Ø2Ø     CONTINUE                                                       484Ø
 1Ø3Ø   CONTINUE                                                         485Ø
 1Ø4Ø   S = IP(N, X(1,K), X(1,K))                                        486Ø
        T = S + T                                                        487Ø
        IF (S - .1E-Ø1*T) 1Ø6Ø, 1Ø6Ø, 1Ø5Ø                               488Ø
 1Ø5Ø   IF (T - MT) 1Ø6Ø, 1Ø6Ø, 1Ø8Ø                                     489Ø
 1Ø6Ø   ORIG = .FALSE.                                                   49ØØ
        IF (S - MT) 1Ø7Ø, 1Ø7Ø, 1ØØØ                                     491Ø
 1Ø7Ø   S = .ØE+ØØ                                                       492Ø
 1Ø8Ø   S = SQRT(S)                                                      493Ø
        WK(K,K,1) = S                                                    494Ø
        IF (S) 1Ø9Ø, 11ØØ, 1Ø9Ø                                          495Ø
 1Ø9Ø   S = .1E+Ø1/S                                                     496Ø
 11ØØ   DO 111Ø J = 1, N                                                 497Ø
          X(J,K) = S*X(J,K)                                              498Ø
 111Ø   CONTINUE                                                         499Ø
 112Ø CONTINUE                                                           5ØØØ
      GO TO IK, (6Ø, 9Ø, 26Ø, 91Ø, 114Ø)                                 5Ø1Ø
C          STATEMENT 113Ø IS EX.                                         5Ø2Ø
 113Ø EM = G                                                             5Ø3Ø
C        SOLVE EIGENVALUE PROBLEM OF PROJECTION OF MATRIX C.             5Ø4Ø
      ASSIGN 114Ø TO IK                                                  5Ø5Ø
      LF = 1                                                             5Ø6Ø
      L1 = JP                                                            5Ø7Ø
      GO TO 99Ø                                                          5Ø8Ø
 114Ø DO 116Ø K = 1, JP                                                  5Ø9Ø
        CALL OP(N, X(1,K), X(1,P))                                       51ØØ
```

```
        DO 115Ø I = 1, K                                                    511Ø
          WK(K,I,1) = -IP(N, X(1,I), X(1,P))                                512Ø
 115Ø   CONTINUE                                                            513Ø
 116Ø CONTINUE                                                              514Ø
      CALL TRED2(P, JP, WK, D, WK(1,4,2), WK)                               515Ø
      CALL IMTQL2(P, JP, D, WK(1,4,2), WK, L)                               516Ø
      WK(IG,2,2) = AMAX1(WK(IG,2,2), .1E+Ø4*FLOAT(L))                       517Ø
C        ARRANGE EIGENVALUES IN ORDER OF DECREASING ABSOLUTE VALUE.         518Ø
      DO 121Ø J = 1, JP                                                     519Ø
        K = J                                                               52ØØ
        DO 117Ø I = J, JP                                                   521Ø
          IF (ABS(D(I)) .GT. ABS(D(K))) K = I                               522Ø
 117Ø   CONTINUE                                                            523Ø
        IF (K - J) 12ØØ, 12ØØ, 118Ø                                         524Ø
 118Ø   T = D(K)                                                            525Ø
        D(K) = D(J)                                                         526Ø
        D(J) = T                                                            527Ø
        DO 119Ø I = 1, JP                                                   528Ø
          T = WK(I,K,1)                                                     529Ø
          WK(I,K,1) = WK(I,J,1)                                             53ØØ
          WK(I,J,1) = T                                                     531Ø
 119Ø   CONTINUE                                                            532Ø
 12ØØ   D(J) = -D(J)                                                        533Ø
 121Ø CONTINUE                                                              534Ø
      DO 125Ø J = 1, N                                                      535Ø
        DO 123Ø I = 1, JP                                                   536Ø
          S = .ØE+ØØ                                                        537Ø
          DO 122Ø K = 1, JP                                                 538Ø
            S = S + X(J,K)*WK(K,I,1)                                        539Ø
 122Ø     CONTINUE                                                          54ØØ
          WK(I,4,2) = S                                                     541Ø
 123Ø   CONTINUE                                                            542Ø
        DO 124Ø I = 1, JP                                                   543Ø
          X(J,I) = WK(I,4,2)                                                544Ø
 124Ø   CONTINUE                                                            545Ø
 125Ø CONTINUE                                                              546Ø
      KM = KS                                                               547Ø
      D(P) = E                                                              548Ø
      RETURN                                                                549Ø
      END                                                                   55ØØ
C     PROGRAM TESTB(INPUT, OUTPUT, TAPE5 = INPUT, TAPE6 = OUTPUT)           551Ø
      DIMENSION A(5Ø,6), T(5Ø,6), Y(5Ø)                                     552Ø
      COMMON NT, T, NA, A, ND, Y                                            553Ø
      INTEGER Y                                                             554Ø
      DIMENSION X(5Ø,1Ø), D(1Ø), WK(115,2)                                  555Ø
      EXTERNAL INTROS, ABAND, BAND                                          556Ø
C                                                                           557Ø
C  THE VARIABLE ND MUST IDENTIFY A VALUE AT LEAST EQUAL TO N.               558Ø
C  THE VALUE ASSIGNED TO THE VARIABLE ND MUST AGREE WITH THE LEADING        559Ø
C  DIMENSION OF THE VARIABLES A, T, X, AND Y.  THE LEADING DIMENSION OF     56ØØ
C  THE VARIABLE WK MUST EQUAL OR EXCEED ND + 65.  THE SECOND DIMENSION      561Ø
C  OF A AND T MUST EXCEED NA AND NT, RESPECTIVELY.  OTHERWISE RECOMPILE.    562Ø
C  NOTE THAT DIMENSIONS OF THE VARIABLES A, T, AND Y MUST AGREE WITH        563Ø
C  THEIR COUNTERPARTS IN FUNCTION BAND AND SUBROUTINE ABAND.                564Ø
C                                                                           565Ø
C  N       IS AN INTEGER INPUT VARIABLE, THE ORDER OF THE MATRICES A AND     566Ø
C          B.  IF N .LE. Ø, PROCESSING CEASES.                              567Ø
C  NA      IS AN INTEGER INPUT VARIABLE, THE NUMBER OF ADJACENT NON-ZERO     568Ø
C          DIAGONALS STRICTLY BELOW THE MAIN DIAGONAL OF THE MATRIX A.       569Ø
C  NT      IS AN INTEGER INPUT VARIABLE, THE NUMBER OF ADJACENT NON-ZERO     57ØØ
C          DIAGONALS STRICTLY BELOW THE MAIN DIAGONAL OF THE MATRIX T.       571Ø
C  KM      IS AN INTEGER INPUT VARIABLE, THE MAXIMUM NUMBER OF ITERATION     572Ø
C          STEPS TO BE EXECUTED BY SIMITZ ON EACH CALL.  IF KM IDENTIFIES    573Ø
C          A NEGATIVE VALUE, SIMITZ WILL USE PREVIOUSLY COMPUTED EIGENVEC    574Ø
C          TORS AS INITIAL APPROXIMATIONS FOR A GIVEN VALUE OF P WHILE EM    575Ø
C          IS INCREASED.                                                     576Ø
C  EPS     IS A REAL INPUT VARIABLE, THE TOLERANCE FOR ACCEPTING EIGEN-      577Ø
C          VECTORS.                                                          578Ø
C  SEED    IS A REAL INPUT VARIABLE TO INITIALIZE THE RANDOM NUMBER GEN-     579Ø
C          ERATOR RANF USING THE INITIALIZING SUBROUTINE RANSET. IF SEED     58ØØ
C          IDENTIFIES A NEGATIVE VALUE, NO PRINTING OF THE NON-ZERO          581Ø
C          DIAGONALS WILL BE DONE.                                           582Ø
C  TRANSA  IS A REAL INPUT VARIABLE, THE LEFT ENDPOINT OF THE INTERVAL       583Ø
C          FROM WHICH THE NON-ZERO ENTRIES OF A AND T ARE SELECTED.          584Ø
C  DIALA   IS A REAL INPUT VARIABLE, A POSITIVE NUMBER SELECTED SO THAT      585Ø
C          TRANSA + DIALA IS THE RIGHT ENDPOINT OF THE INTERVAL OVER         586Ø
```

```
C         WHICH THE NON-ZERO ENTRIES OF A AND T ARE SELECTED.                  587Ø
C                                                                              588Ø
      ND = 5Ø                                                                  589Ø
 1Ø   READ (5, 2Ø) N, NA, NT, KM, EPS, SEED, TRANSA, DIALA                     59ØØ
 2Ø   FORMAT(4I5,1P4E1Ø.2)                                                     591Ø
      IF (N .LE. Ø) STOP                                                       592Ø
      CALL RANSET(ABS(SEED))                                                   593Ø
      MP1 = NT + 1                                                             594Ø
      DO 4Ø I = 1, N                                                           595Ø
        MAX = MAXØ(1, -I + (MP1 + 1))                                          596Ø
        DO 3Ø J = MAX, MP1                                                     597Ø
          T(I,J) = AINT(TRANSA + DIALA*RANF(1))                                598Ø
 3Ø     CONTINUE                                                               599Ø
        T(I,MP1) = AMAX1(ABS(T(I,MP1)), ABS(AINT(TRANSA + DIALA)) -            6ØØØ
     *                   ABS(T(I,MP1)))                                        6Ø1Ø
 4Ø   CONTINUE                                                                 6Ø2Ø
      WRITE (6, 5Ø) N, NA, NT, KM, EPS, SEED, TRANSA, DIALA                    6Ø3Ø
 5Ø   FORMAT(1H1,6X,1HN,11X,2HNA,11X,2HNB,11X,2HKM,18X,3HEPS,17X,4HSEED,       6Ø4Ø
     *       1ØX,11HTRANSLATION,13X,8HDILATION/1H /I8,3I13,1P4E21.2/1HØ)       6Ø5Ø
      IF (SEED) 11Ø, 6Ø, 6Ø                                                    6Ø6Ø
 6Ø   WRITE (6, 7Ø)                                                           6Ø7Ø
 7Ø   FORMAT(1HØ,36X,59HADJACENT NON-ZERO LOWER DIAGONALS OF T, T(T-TRAN      6Ø8Ø
     *SPOSED) = B)                                                            6Ø9Ø
      MAX = MP1                                                                61ØØ
      DO 9Ø J = 1, MP1                                                         611Ø
        DO 8Ø I = J, N                                                         612Ø
          Y(I) = IFIX(T(I,MAX))                                                613Ø
 8Ø     CONTINUE                                                               614Ø
        WRITE (6, 1ØØ) (Y(I), I = J, N)                                        615Ø
        MAX = MAX - 1                                                          616Ø
 9Ø   CONTINUE                                                                 617Ø
 1ØØ  FORMAT(1H /(16X,15I7))                                                   618Ø
 11Ø  MP1 = NA + 1                                                             619Ø
      DO 13Ø I = 1, N                                                          62ØØ
        MAX = MAXØ(1, -I + (MP1 + 1))                                          621Ø
        DO 12Ø J = MAX, MP1                                                    622Ø
          A(I,J) = AINT(TRANSA + DIALA*RANF(1))                                623Ø
 12Ø    CONTINUE                                                               624Ø
 13Ø  CONTINUE                                                                 625Ø
      IF (SEED) 18Ø, 14Ø, 14Ø                                                  626Ø
 14Ø  WRITE (6, 15Ø)                                                           627Ø
 15Ø  FORMAT(1HØ/39X,53HADJACENT NON-ZERO LOWER DIAGONALS OF A = A-TRANS      628Ø
     *POSED)                                                                  629Ø
      MAX = MP1                                                                63ØØ
      DO 17Ø J = 1, MP1                                                        631Ø
        DO 16Ø I = J, N                                                        632Ø
          Y(I) = IFIX(A(I,MAX))                                                633Ø
 16Ø    CONTINUE                                                               634Ø
        WRITE (6, 1ØØ) (Y(I), I = J, N)                                        635Ø
        MAX = MAX - 1                                                          636Ø
 17Ø  CONTINUE                                                                 637Ø
 18Ø  WRITE (6, 19Ø)                                                           638Ø
 19Ø  FORMAT(1HØ/1HØ,21X,1HP,5X,6HEM(IN),4X,7HEM(OUT),9X,2HKS,1ØX,1HK,        639Ø
     *6X,12HMAX RESIDUAL,5X,13HMEAN RESIDUAL,8X,1ØHTIME(SECS)/1H )            64ØØ
      MP = MINØ(N/5, 1Ø)                                                       641Ø
      IF (KM) 192, 198, 198                                                    642Ø
 192  DO 196 J = 1, MP                                                         643Ø
        DO 194 I = 1, N                                                        644Ø
          X(I,J) = 2.Ø*RANF(2.Ø) - 1.Ø                                         645Ø
 194    CONTINUE                                                               646Ø
 196  CONTINUE                                                                 647Ø
 198  DO 28Ø IP = 2, MP                                                        648Ø
        MAXM = IP - 1                                                          649Ø
        DO 27Ø M = 1, MAXM                                                     65ØØ
          MM = M                                                               651Ø
          KS = KM                                                              652Ø
          T1 = SECOND(S)                                                       653Ø
          CALL SIMITZ(N, IP, KS, EPS, BAND, ABAND, INTROS, MM, X, ND, D,       654Ø
     *                WK)                                                     655Ø
          T1 = SECOND(S) - T1                                                  656Ø
          IF (MM) 2ØØ, 2ØØ, 21Ø                                               657Ø
 2ØØ      L = Ø                                                                658Ø
          ANMAX = 1.Ø                                                          659Ø
          ANMEAN = 1.Ø                                                         66ØØ
          GO TO 25Ø                                                            661Ø
 21Ø      PROD = 1.Ø                                                           662Ø
```

```
          ANMAX = Ø.Ø                                                          663Ø
          L = 1                                                                664Ø
 22Ø      DO 24Ø J = 1, MM                                                     665Ø
            CALL BANDOP(N, NA, A, ND, X(1,J), WK(1,1))                         666Ø
            CALL TANDOP(N, NT, T, ND, X(1,J), WK(1,2))                         667Ø
            S = Ø.Ø                                                            668Ø
            DO 23Ø I = 1, N                                                    669Ø
              S = S + (WK(I,1) - D(J)*WK(I,2))*(WK(I,1) - D(J)*WK(I,2))        67ØØ
 23Ø        CONTINUE                                                           671Ø
            PROD = PROD*S                                                      672Ø
            ANMAX = AMAX1(ANMAX, S)                                            673Ø
            L = MAX1(FLOAT(L), SIGN(FLOAT(J), S - ANMAX))                      674Ø
 24Ø      CONTINUE                                                             675Ø
          S = ABS(D(1)) - D(IP)                                                676Ø
          ANMAX = SQRT(ANMAX)/S                                                677Ø
          ANMEAN = SQRT(PROD**(1.Ø/FLOAT(MM)))/S                               678Ø
 25Ø      WRITE (6,26Ø) IP, M, MM, KS, L, ANMAX, ANMEAN, T1                    679Ø
 26Ø      FORMAT(I23,4I11,1P2E18.2,ØPF18.2)                                    68ØØ
 27Ø    CONTINUE                                                               681Ø
 28Ø  CONTINUE                                                                 682Ø
      WRITE (6,29Ø) (D(J), J = 1, MAXM)                                        683Ø
 29Ø  FORMAT(1HØ/1HØ,57X,17HFINAL EIGENVALUES/1H /(1PE37.12,4E21.12))          684Ø
      GO TO 1Ø                                                                 685Ø
      END                                                                      686Ø


      SUBROUTINE INTROS(KS, G, H, F)                                           687Ø
C  INTROS IS A DUMMY SUBROUTINE INF.                                           688Ø
      INTEGER G, H                                                             689Ø
      REAL F(1)                                                                69ØØ
      RETURN                                                                   691Ø
      END                                                                      692Ø


      SUBROUTINE BANDOP(N, NA, A, ND, V, W)                                    693Ø
C  CALCULATE THE IMAGE W OF THE N-VECTOR V UNDER                               694Ø
C  THE MATRIX A WHERE A IS SYMMETRIC AND BANDED OF BAND WIDTH 2*NA + 1.        695Ø
      DIMENSION A(ND,1), V(1), W(1)                                            696Ø
      INTEGER P, Q, R                                                          697Ø
      MP1 = NA + 1                                                             698Ø
      DO 7Ø I = 1, N                                                           699Ø
        P = MAXØ(1, -I + (MP1 + 1))                                            7ØØØ
        Q = MAXØ(1, I + (MP1 - N))                                             7Ø1Ø
        T = A(I,MP1)*V(I)                                                      7Ø2Ø
        IF (NA - P) 3Ø, 1Ø, 1Ø                                                 7Ø3Ø
 1Ø     R = I - (MP1 - P)                                                      7Ø4Ø
        DO 2Ø K = P, NA                                                        7Ø5Ø
          T = T + A(I,K)*V(R)                                                  7Ø6Ø
          R = R + 1                                                            7Ø7Ø
 2Ø     CONTINUE                                                               7Ø8Ø
 3Ø     IF (NA - Q) 6Ø, 4Ø, 4Ø                                                 7Ø9Ø
 4Ø     R = I + (MP1 - Q)                                                      71ØØ
        DO 5Ø K = Q, NA                                                        711Ø
          T = T + A(R,K)*V(R)                                                  712Ø
          R = R - 1                                                            713Ø
 5Ø     CONTINUE                                                               714Ø
 6Ø     W(I) = T                                                               715Ø
 7Ø   CONTINUE                                                                 716Ø
      RETURN                                                                   717Ø
      END                                                                      718Ø


      SUBROUTINE TANDOP(N, NA, A, ND, V, W)                                    719Ø
C  COMPUTE THE IMAGE W OF THE N-VECTOR V UNDER THE N-SQUARE MATRIX             72ØØ
C    A(A-TRANSPOSED), A LOWER TRIANGULAR OF BANDWIDTH NA + 1.                  721Ø
      DIMENSION A(ND,1), V(1), W(1)                                            722Ø
      INTEGER P, Q, R                                                          723Ø
      MP1 = NA + 1                                                             724Ø
      DO 4Ø I = 1, N                                                           725Ø
        P = MAXØ(1, I + (MP1 - N))                                             726Ø
        T = A(I,MP1)*V(I)                                                      727Ø
        IF (NA - P) 3Ø, 1Ø, 1Ø                                                 728Ø
 1Ø     R = I + (MP1 - P)                                                      729Ø
        DO 2Ø K = P, NA                                                        73ØØ
          T = T + A(R,K)*V(R)                                                  731Ø
          R = R - 1                                                            732Ø
```

```
2Ø      CONTINUE                                                        733Ø
3Ø      W(I) = T                                                        734Ø
4Ø    CONTINUE                                                          735Ø
      I = N                                                             736Ø
      DO 8Ø Q = 1, N                                                    737Ø
        P = MAXØ(1, -I + (MP1 + 1))                                     738Ø
        T = A(I,MP1)*W(I)                                               739Ø
        IF (NA - P) 7Ø, 5Ø, 5Ø                                          74ØØ
5Ø      R = I - (MP1 - P)                                               741Ø
        DO 6Ø K = P, NA                                                 742Ø
          T = T + A(I,K)*W(R)                                           743Ø
          R = R + 1                                                     744Ø
6Ø      CONTINUE                                                        745Ø
7Ø      W(I) = T                                                        746Ø
        I = I - 1                                                       747Ø
8Ø    CONTINUE                                                          748Ø
      RETURN                                                            749Ø
      END                                                               75ØØ


      FUNCTION BAND(N, Z, W)                                            751Ø
C  CALCULATE THE INNER PRODUCT (W-TRANSPOSED)BZ WHERE B =               752Ø
C       T(T-TRANSPOSED).                                                753Ø
      DIMENSION A(5Ø,6), T(5Ø,6), Y(5Ø)                                 754Ø
      COMMON NT, T, NA, A, ND, Y                                        755Ø
      DIMENSION W(1), Z(1)                                              756Ø
 1Ø   CALL TANDOP(N, NT, T, ND, Z, Y)                                   757Ø
      S = Ø.Ø                                                           758Ø
      DO 15 I = 1, N                                                    759Ø
        S = S + W(I)*Y(I)                                               76ØØ
 15   CONTINUE                                                          761Ø
      BAND = S                                                          762Ø
      RETURN                                                            763Ø
      END                                                               764Ø


      SUBROUTINE ABAND(N, Z, W)                                         765Ø
C  CALCULATE THE SOLUTION W OF THE LINEAR SYSTEM BW = AZ GIVEN THE       766Ø
C  N-VECTOR Z, THE BANDED SYMMETRIC N-SQUARE MATRIX A OF BAND WIDTH      767Ø
C  2*NA + 1, AND THE BANDED TRIANGULAR FACTOR T OF BAND WIDTH NT + 1     768Ø
C  OF THE POSITIVE DEFINITE MATRIX B, T(T-TRANSPOSED) = B.               769Ø
      DIMENSION W(1), Z(1)                                              77ØØ
      DIMENSION A(5Ø,6), T(5Ø,6), Y(5Ø)                                 771Ø
      COMMON NT, T, NA, A, ND, Y                                        772Ø
      CALL BANDOP(N, NA, A, ND, Z, W)                                   773Ø
      CALL SOLVE(N, NT, T, ND, W)                                       774Ø
      RETURN                                                            775Ø
      END                                                               776Ø


      SUBROUTINE SOLVE(N, M, A, N1, B)                                  777Ø
C  SOLVE A LINEAR SYSTEM GIVEN A TRIANGULAR FACTORIZATION OF A BANDED    778Ø
C    POSITIVE DEFINITE COEFFICIENT MATRIX OF BAND WIDTH 2*M + 1.         779Ø
      DIMENSION A(N1,1), B(N)                                           78ØØ
      INTEGER P, Q, R                                                   781Ø
      MP1 = M + 1                                                       782Ø
COMMENT  SOLUTION OF LY = B                                             783Ø
      DO 13Ø I = 1, N                                                   784Ø
        P = MAXØ(1, MP1 - I + 1)                                        785Ø
        T = B(I)                                                        786Ø
        IF (M - P) 12Ø, 1ØØ, 1ØØ                                        787Ø
 1ØØ    Q = I - MP1 + P                                                 788Ø
        DO 11Ø K = P, M                                                 789Ø
          T = T - A(I,K)*B(Q)                                           79ØØ
          Q = Q + 1                                                     791Ø
 11Ø    CONTINUE                                                        792Ø
 12Ø    B(I) = T/A(I,MP1)                                               793Ø
 13Ø  CONTINUE                                                          794Ø
COMMENT  SOLUTION OF UX = Y                                             795Ø
      I = N                                                             796Ø
      DO 17Ø R = 1, N                                                   797Ø
        P = MAXØ(1, MP1 - N + I)                                        798Ø
        T = B(I)                                                        799Ø
        IF (M - P) 16Ø, 14Ø, 14Ø                                        8ØØØ
 14Ø    Q = I + MP1 - P                                                 8Ø1Ø
        DO 15Ø K = P, M                                                 8Ø2Ø
```

```
            T = T - A(Q,K)*B(Q)                                          8Ø3Ø
            Q = Q - 1                                                    8Ø4Ø
 15Ø      CONTINUE                                                       8Ø5Ø
 16Ø      B(I) = T/A(I,MP1)                                              8Ø6Ø
          I = I - 1                                                      8Ø7Ø
 17Ø    CONTINUE                                                         8Ø8Ø
        RETURN                                                           8Ø9Ø
        END                                                              81ØØ


        FUNCTION RANF(X)                                                 811Ø
C   RETURNS A RANDOM NUMBER ON THE OPEN UNIT INTERVAL GIVEN ANY ARGUMENT. 812Ø
        COMMON /TSEED/ K                                                 813Ø
        K = MOD(3125*K, 65536)                                           814Ø
        RANF = ABS(FLOAT(K - 32768)/32768.Ø)                             815Ø
        RETURN                                                           816Ø
        END                                                              817Ø


        SUBROUTINE RANSET(SEED)                                          818Ø
C   INITIALIZES THE RANDOM NUMBER SEED K USING THE REAL INPUT            819Ø
C   VARIABLE SEED.                                                       82ØØ
        COMMON /TSEED/ K                                                 821Ø
        T = SEED                                                         822Ø
        IF (ABS(SEED) .GT. 1.Ø) T = 1.Ø/T                                823Ø
        K = 2*IFIX(16384.Ø*T) - IFIX(SIGN(1.Ø, T))                       824Ø
        RETURN                                                           825Ø
        END                                                              826Ø


        FUNCTION SECOND(T)                                               827Ø
C   DUMMY FUNCTION SECOND RETURNS THE VALUES SECOND = T = 0.0.           828Ø
C                                                                        829Ø
C   SECOND MAY BE MODIFIED TO CALL AN OPERATING SYSTEM DEPENDENT PROGRAM  83ØØ
C   WHICH RETURNS CENTRAL PROCESSOR TIME IN SECONDS RELATIVE TO          831Ø
C   AN ARBITRARY STARTING POINT AS A REAL VALUE T.                       832Ø
C                                                                        833Ø
        T = Ø.Ø                                                          834Ø
        SECOND = Ø.Ø                                                     835Ø
        RETURN                                                           836Ø
        END                                                              837Ø


C                                                                        838Ø
C     ------------------------------------------------------------------ 839Ø
C                                                                        84ØØ
      SUBROUTINE IMTQL2(NM,N,D,E,Z,IERR)                                 841Ø
C                                                                        842Ø
      INTEGER I,J,K,L,M,N,II,NM,MML,IERR                                 843Ø
      REAL D(N),E(N),Z(NM,N)                                             844Ø
      REAL B,C,F,G,P,R,S,MACHEP                                          845Ø
C     REAL SQRT,ABS,SIGN                                                 846Ø
C                                                                        847Ø
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE IMTQL2,    848Ø
C     NUM. MATH. 12, 377-383(1968) BY MARTIN AND WILKINSON,              849Ø
C     AS MODIFIED IN NUM. MATH. 15, 45Ø(197Ø) BY DUBRULLE.               85ØØ
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 241-248(1971).    851Ø
C                                                                        852Ø
C     THIS SUBROUTINE FINDS THE EIGENVALUES AND EIGENVECTORS             853Ø
C     OF A SYMMETRIC TRIDIAGONAL MATRIX BY THE IMPLICIT QL METHOD.       854Ø
C     THE EIGENVECTORS OF A FULL SYMMETRIC MATRIX CAN ALSO               855Ø
C     BE FOUND IF  TRED2  HAS BEEN USED TO REDUCE THIS                   856Ø
C     FULL MATRIX TO TRIDIAGONAL FORM.                                   857Ø
C                                                                        858Ø
C     ON INPUT-                                                          859Ø
C                                                                        86ØØ
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL          861Ø
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM           862Ø
C          DIMENSION STATEMENT,                                          863Ø
C                                                                        864Ø
C        N IS THE ORDER OF THE MATRIX,                                   865Ø
C                                                                        866Ø
C        D CONTAINS THE DIAGONAL ELEMENTS OF THE INPUT MATRIX,           867Ø
C                                                                        868Ø
C        E CONTAINS THE SUBDIAGONAL ELEMENTS OF THE INPUT MATRIX         869Ø
```

```
C          IN ITS LAST N-1 POSITIONS.  E(1) IS ARBITRARY,                  87ØØ
C                                                                          871Ø
C        Z CONTAINS THE TRANSFORMATION MATRIX PRODUCED IN THE              872Ø
C          REDUCTION BY  TRED2, IF PERFORMED.  IF THE EIGENVECTORS         873Ø
C          OF THE TRIDIAGONAL MATRIX ARE DESIRED, Z MUST CONTAIN           874Ø
C          THE IDENTITY MATRIX.                                            875Ø
C                                                                          876Ø
C      ON OUTPUT-                                                          877Ø
C                                                                          878Ø
C        D CONTAINS THE EIGENVALUES IN ASCENDING ORDER.  IF AN             879Ø
C          ERROR EXIT IS MADE, THE EIGENVALUES ARE CORRECT BUT             88ØØ
C          UNORDERED FOR INDICES 1,2,...,IERR-1,                           881Ø
C                                                                          882Ø
C        E HAS BEEN DESTROYED,                                             883Ø
C                                                                          884Ø
C        Z CONTAINS ORTHONORMAL EIGENVECTORS OF THE SYMMETRIC              885Ø
C          TRIDIAGONAL (OR FULL) MATRIX.  IF AN ERROR EXIT IS MADE,        886Ø
C          Z CONTAINS THE EIGENVECTORS ASSOCIATED WITH THE STORED          887Ø
C          EIGENVALUES,                                                    888Ø
C                                                                          889Ø
C        IERR IS SET TO                                                    89ØØ
C          ZERO       FOR NORMAL RETURN,                                   891Ø
C          J          IF THE J-TH EIGENVALUE HAS NOT BEEN                  892Ø
C                     DETERMINED AFTER 3Ø ITERATIONS.                      893Ø
C                                                                          894Ø
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO B. S. GARBOW,           895Ø
C     APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY            896Ø
C                                                                          897Ø
C     ------------------------------------------------------------------   898Ø
C                                                                          899Ø
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING        9ØØØ
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.      9Ø1Ø
C                                                                          9Ø2Ø
C                **********                                                9Ø3Ø
      MACHEP = 2.**(-47)                                                   9Ø4Ø
C                                                                          9Ø5Ø
      IERR = Ø                                                             9Ø6Ø
      IF (N .EQ. 1) GO TO 1ØØ1                                             9Ø7Ø
C                                                                          9Ø8Ø
      DO 1ØØ I = 2, N                                                      9Ø9Ø
  1ØØ E(I-1) = E(I)                                                        91ØØ
C                                                                          911Ø
      E(N) = Ø.Ø                                                           912Ø
C                                                                          913Ø
      DO 24Ø L = 1, N                                                      914Ø
         J = Ø                                                             915Ø
C     ********** LOOK FOR SMALL SUB-DIAGONAL ELEMENT **********            916Ø
  1Ø5    DO 11Ø M = L, N                                                   917Ø
            IF (M .EQ. N) GO TO 12Ø                                        918Ø
            IF (ABS(E(M)) .LE. MACHEP * (ABS(D(M)) + ABS(D(M+1))))         919Ø
     X         GO TO 12Ø                                                   92ØØ
  11Ø    CONTINUE                                                          921Ø
C                                                                          922Ø
  12Ø    P = D(L)                                                          923Ø
         IF (M .EQ. L) GO TO 24Ø                                           924Ø
         IF (J .EQ. 3Ø) GO TO 1ØØØ                                         925Ø
         J = J + 1                                                         926Ø
C     ********** FORM SHIFT **********                                     927Ø
         G = (D(L+1) - P) / (2.Ø * E(L))                                   928Ø
         R = SQRT(G*G+1.Ø)                                                 929Ø
         G = D(M) - P + E(L) / (G + SIGN(R,G))                             93ØØ
         S = 1.Ø                                                           931Ø
         C = 1.Ø                                                           932Ø
         P = Ø.Ø                                                           933Ø
         MML = M - L                                                       934Ø
C     ********** FOR I=M-1 STEP -1 UNTIL L DO -- **********               935Ø
         DO 2ØØ II = 1, MML                                                936Ø
            I = M - II                                                     937Ø
            F = S * E(I)                                                   938Ø
            B = C * E(I)                                                   939Ø
            IF (ABS(F) .LT. ABS(G)) GO TO 15Ø                              94ØØ
            C = G / F                                                      941Ø
            R = SQRT(C*C+1.Ø)                                              942Ø
            E(I+1) = F * R                                                 943Ø
            S = 1.Ø / R                                                    944Ø
            C = C * S                                                      945Ø
```

```
            GO TO 16Ø                                                    946Ø
   15Ø      S = F / G                                                    947Ø
            R = SQRT(S*S+1.Ø)                                            948Ø
            E(I+1) = G * R                                               949Ø
            C = 1.Ø / R                                                  95ØØ
            S = S * C                                                    951Ø
   16Ø      G = D(I+1) - P                                               952Ø
            R = (D(I) - G) * S + 2.Ø * C * B                             953Ø
            P = S * R                                                    954Ø
            D(I+1) = G + P                                               955Ø
            G = C * R - B                                                956Ø
C     ********** FORM VECTOR **********                                  957Ø
            DO 18Ø K = 1, N                                              958Ø
               F = Z(K,I+1)                                              959Ø
               Z(K,I+1) = S * Z(K,I) + C * F                             96ØØ
               Z(K,I) = C * Z(K,I) - S * F                               961Ø
   18Ø      CONTINUE                                                     962Ø
C                                                                       963Ø
   2ØØ   CONTINUE                                                       964Ø
C                                                                       965Ø
         D(L) = D(L) - P                                                966Ø
         E(L) = G                                                       967Ø
         E(M) = Ø.Ø                                                     968Ø
         GO TO 1Ø5                                                      969Ø
   24Ø CONTINUE                                                         97ØØ
C     ********** ORDER EIGENVALUES AND EIGENVECTORS **********          971Ø
      DO 3ØØ II = 2, N                                                  972Ø
         I = II - 1                                                     973Ø
         K = I                                                          974Ø
         P = D(I)                                                       975Ø
C                                                                       976Ø
         DO 26Ø J = II, N                                               977Ø
            IF (D(J) .GE. P) GO TO 26Ø                                  978Ø
            K = J                                                       979Ø
            P = D(J)                                                    98ØØ
   26Ø   CONTINUE                                                       981Ø
C                                                                       982Ø
         IF (K .EQ. I) GO TO 3ØØ                                        983Ø
         D(K) = D(I)                                                    984Ø
         D(I) = P                                                       985Ø
C                                                                       986Ø
         DO 28Ø J = 1, N                                                987Ø
            P = Z(J,I)                                                  988Ø
            Z(J,I) = Z(J,K)                                             989Ø
            Z(J,K) = P                                                  99ØØ
   28Ø   CONTINUE                                                       991Ø
C                                                                       992Ø
   3ØØ CONTINUE                                                         993Ø
C                                                                       994Ø
      GO TO 1ØØ1                                                        995Ø
C     ********** SET ERROR -- NO CONVERGENCE TO AN                       996Ø
C                EIGENVALUE AFTER 3Ø ITERATIONS **********              997Ø
  1ØØØ IERR = L                                                         998Ø
  1ØØ1 RETURN                                                           999Ø
C     ********** LAST CARD OF IMTQL2 **********                        1ØØØØ
      END                                                              1ØØ1Ø
C                                                                      1ØØ2Ø
C     ------------------------------------------------------------     1ØØ3Ø
C                                                                      1ØØ4Ø
      SUBROUTINE TRED2(NM,N,A,D,E,Z)                                   1ØØ5Ø
C                                                                      1ØØ6Ø
      INTEGER I,J,K,L,N,II,NM,JP1                                      1ØØ7Ø
      REAL A(NM,N),D(N),E(N),Z(NM,N)                                   1ØØ8Ø
      REAL F,G,H,HH,SCALE                                              1ØØ9Ø
C     REAL SQRT,ABS,SIGN                                               1Ø1ØØ
C                                                                      1Ø11Ø
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE TRED2,   1Ø12Ø
C     NUM. MATH. 11, 181-195(1968) BY MARTIN, REINSCH, AND WILKINSON.  1Ø13Ø
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 212-226(1971).  1Ø14Ø
C                                                                      1Ø15Ø
C     THIS SUBROUTINE REDUCES A REAL SYMMETRIC MATRIX TO A             1Ø16Ø
C     SYMMETRIC TRIDIAGONAL MATRIX USING AND ACCUMULATING              1Ø17Ø
C     ORTHOGONAL SIMILARITY TRANSFORMATIONS.                           1Ø18Ø
C                                                                      1Ø19Ø
C     ON INPUT-                                                        1Ø2ØØ
```

```
C                                                                       1Ø21Ø
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL         1Ø22Ø
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM          1Ø23Ø
C          DIMENSION STATEMENT,                                         1Ø24Ø
C                                                                       1Ø25Ø
C        N IS THE ORDER OF THE MATRIX,                                  1Ø26Ø
C                                                                       1Ø27Ø
C        A CONTAINS THE REAL SYMMETRIC INPUT MATRIX.  ONLY THE          1Ø28Ø
C          LOWER TRIANGLE OF THE MATRIX NEED BE SUPPLIED.               1Ø29Ø
C                                                                       1Ø3ØØ
C     ON OUTPUT-                                                        1Ø31Ø
C                                                                       1Ø32Ø
C        D CONTAINS THE DIAGONAL ELEMENTS OF THE TRIDIAGONAL MATRIX,    1Ø33Ø
C                                                                       1Ø34Ø
C        E CONTAINS THE SUBDIAGONAL ELEMENTS OF THE TRIDIAGONAL         1Ø35Ø
C          MATRIX IN ITS LAST N-1 POSITIONS.  E(1) IS SET TO ZERO,      1Ø36Ø
C                                                                       1Ø37Ø
C        Z CONTAINS THE ORTHOGONAL TRANSFORMATION MATRIX                1Ø38Ø
C          PRODUCED IN THE REDUCTION,                                   1Ø39Ø
C                                                                       1Ø4ØØ
C        A AND Z MAY COINCIDE.  IF DISTINCT, A IS UNALTERED.            1Ø41Ø
C                                                                       1Ø42Ø
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO B. S. GARBOW,        1Ø43Ø
C     APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY         1Ø44Ø
C                                                                       1Ø45Ø
C     ------------------------------------------------------------------1Ø46Ø
C                                                                       1Ø47Ø
      DO 1ØØ I = 1, N                                                   1Ø48Ø
C                                                                       1Ø49Ø
         DO 1ØØ J = 1, I                                                1Ø5ØØ
            Z(I,J) = A(I,J)                                             1Ø51Ø
  1ØØ CONTINUE                                                          1Ø52Ø
C                                                                       1Ø53Ø
      IF (N .EQ. 1) GO TO 32Ø                                           1Ø54Ø
C     ********** FOR I=N STEP -1 UNTIL 2 DO -- **********               1Ø55Ø
      DO 3ØØ II = 2, N                                                  1Ø56Ø
         I = N + 2 - II                                                 1Ø57Ø
         L = I - 1                                                      1Ø58Ø
         H = Ø.Ø                                                        1Ø59Ø
         SCALE = Ø.Ø                                                    1Ø6ØØ
         IF (L .LT. 2) GO TO 13Ø                                        1Ø61Ø
C     ********** SCALE ROW (ALGOL TOL THEN NOT NEEDED) **********       1Ø62Ø
         DO 12Ø K = 1, L                                                1Ø63Ø
  12Ø    SCALE = SCALE + ABS(Z(I,K))                                    1Ø64Ø
C                                                                       1Ø65Ø
         IF (SCALE .NE. Ø.Ø) GO TO 14Ø                                  1Ø66Ø
  13Ø    E(I) = Z(I,L)                                                  1Ø67Ø
         GO TO 29Ø                                                      1Ø68Ø
C                                                                       1Ø69Ø
  14Ø    DO 15Ø K = 1, L                                                1Ø7ØØ
            Z(I,K) = Z(I,K) / SCALE                                     1Ø71Ø
            H = H + Z(I,K) * Z(I,K)                                     1Ø72Ø
  15Ø    CONTINUE                                                       1Ø73Ø
C                                                                       1Ø74Ø
         F = Z(I,L)                                                     1Ø75Ø
         G = -SIGN(SQRT(H),F)                                           1Ø76Ø
         E(I) = SCALE * G                                               1Ø77Ø
         H = H - F * G                                                  1Ø78Ø
         Z(I,L) = F - G                                                 1Ø79Ø
         F = Ø.Ø                                                        1Ø8ØØ
C                                                                       1Ø81Ø
         DO 24Ø J = 1, L                                                1Ø82Ø
            Z(J,I) = Z(I,J) / H                                         1Ø83Ø
            G = Ø.Ø                                                     1Ø84Ø
C     ********** FORM ELEMENT OF A*U **********                        1Ø85Ø
            DO 18Ø K = 1, J                                             1Ø86Ø
  18Ø       G = G + Z(J,K) * Z(I,K)                                     1Ø87Ø
C                                                                       1Ø88Ø
            JP1 = J + 1                                                 1Ø89Ø
            IF (L .LT. JP1) GO TO 22Ø                                   1Ø9ØØ
C                                                                       1Ø91Ø
            DO 2ØØ K = JP1, L                                           1Ø92Ø
  2ØØ       G = G + Z(K,J) * Z(I,K)                                     1Ø93Ø
C     ********** FORM ELEMENT OF P **********                          1Ø94Ø
  22Ø       E(J) = G / H                                                1Ø95Ø
            F = F + E(J) * Z(I,J)                                       1Ø96Ø
```

```
  24Ø    CONTINUE                                                        1Ø97Ø
C                                                                        1Ø98Ø
       HH = F / (H + H)                                                  1Ø99Ø
C     ********** FORM REDUCED A **********                               11ØØØ
       DO 26Ø J = 1, L                                                   11Ø1Ø
          F = Z(I,J)                                                     11Ø2Ø
          G = E(J) - HH * F                                              11Ø3Ø
          E(J) = G                                                       11Ø4Ø
C                                                                        11Ø5Ø
          DO 26Ø K = 1, J                                                11Ø6Ø
             Z(J,K) = Z(J,K) - F * E(K) - G * Z(I,K)                     11Ø7Ø
  26Ø    CONTINUE                                                        11Ø8Ø
C                                                                        11Ø9Ø
  29Ø    D(I) = H                                                        111ØØ
  3ØØ CONTINUE                                                           1111Ø
C                                                                        1112Ø
  32Ø D(1) = Ø.Ø                                                         1113Ø
      E(1) = Ø.Ø                                                         1114Ø
C     ********** ACCUMULATION OF TRANSFORMATION MATRICES **********      1115Ø
      DO 5ØØ I = 1, N                                                    1116Ø
         L = I - 1                                                       1117Ø
         IF (D(I) .EQ. Ø.Ø) GO TO 38Ø                                    1118Ø
C                                                                        1119Ø
         DO 36Ø J = 1, L                                                 112ØØ
            G = Ø.Ø                                                      1121Ø
C                                                                        1122Ø
            DO 34Ø K = 1, L                                              1123Ø
  34Ø       G = G + Z(I,K) * Z(K,J)                                      1124Ø
C                                                                        1125Ø
            DO 36Ø K = 1, L                                              1126Ø
               Z(K,J) = Z(K,J) - G * Z(K,I)                              1127Ø
  36Ø    CONTINUE                                                        1128Ø
C                                                                        1129Ø
  38Ø    D(I) = Z(I,I)                                                   113ØØ
         Z(I,I) = 1.Ø                                                    1131Ø
         IF (L .LT. 1) GO TO 5ØØ                                         1132Ø
C                                                                        1133Ø
         DO 4ØØ J = 1, L                                                 1134Ø
            Z(I,J) = Ø.Ø                                                 1135Ø
            Z(J,I) = Ø.Ø                                                 1136Ø
  4ØØ    CONTINUE                                                        1137Ø
C                                                                        1138Ø
  5ØØ CONTINUE                                                           1139Ø
C                                                                        114ØØ
      RETURN                                                             1141Ø
C     ********** LAST CARD OF TRED2 **********                          1142Ø
      END                                                                1143Ø
```

```
   3Ø    5    4  2ØØ  1.ØØE-1Ø  1.ØØE-Ø1 -1.ØØE+Ø1  2.ØØE+Ø1            DATA
   3Ø    5    4 -2ØØ  1.ØØE-1Ø -1.ØØE-Ø1 -1.ØØE+Ø1  2.ØØE+Ø1            DATA
   4Ø    4    3  2ØØ  1.ØØE-1Ø  2.ØØE-Ø1 -1.ØØE+Ø2  2.ØØE+Ø2            DATA
   4Ø    4    3 -2ØØ  1.ØØE-1Ø -2.ØØE-Ø1 -1.ØØE+Ø2  2.ØØE+Ø2            DATA
   5Ø    2    1  2ØØ  1.ØØE-1Ø  3.ØØE-Ø1 -1.ØØE+Ø2  2.ØØE+Ø2            DATA
   5Ø    2    1 -2ØØ  1.ØØE-1Ø -3.ØØE-Ø1 -1.ØØE+Ø2  2.ØØE+Ø2            DATA
    Ø                                                                    DATA
1      N           NA           NB           KM                EPS

      3Ø           5            4          2ØØ           1.ØØE-1Ø

Ø
Ø                                                  ADJACENT NON-ZERO LOWER DIAGONALS

                    7      7      7      5     1Ø      9     1Ø      6      9
                    5      9      7      7      5      8      7      6      8

                    8      5     -5      7     -5      3     -4      8     -9
                    5      6     -4     -5     -9     -7      2      5     -3

                    4      5      3      3     -6      9     -6     -6      2
                    Ø     -2      2      7      6     -1      4      6     -1

                    Ø     -7     -2      Ø     -9      4      4     -9      8
                   -8     -7      Ø      4      6     -8     -5     -3     -5

                   -6      9      6     -4      3      7      2      1     -1
                    1     -7     -8      8      9     -5      3      3      5
```

ADJACENT NON-ZERO LOWER DIAGONALS

Ø	7	6	-4	-6	5	8	3	-7
-8	1	-4	6	-9	8	5	3	-4
-7	-1	Ø	4	8	1	-5	Ø	5
-4	Ø	4	4	5	-2	2	-7	-3
-2	-1	-8	7	1	Ø	-6	6	9
2	1	9	5	-6	-6	-6	Ø	-9
Ø	Ø	-8	Ø	-9	Ø	Ø	7	6
6	5	-6	9	-8	5	-4	-9	6
-7	Ø	-5	8	Ø	-6	-9	5	-7
6	-5	2	Ø	6	-4	-5	-2	-4
-8	Ø	8	3	9	4	1	9	-8
3	3	-8	Ø	2	-6	-4	Ø	4

Ø
Ø

P	EM(IN)	EM(OUT)	KS	K	MAX RES
2	1	1	29	1	8.6
3	1	1	15	1	6.4
3	2	2	24	1	4.9
4	1	1	15	1	1.1
4	2	2	15	1	1.3
4	3	3	51	1	1.2
5	1	1	13	1	9.Ø
5	2	2	13	1	1.1
5	3	3	21	1	6.2
5	4	4	21	1	1.3
6	1	1	13	1	1.7
6	2	2	13	1	5.4
6	3	3	21	1	7.2
6	4	4	13	1	9.6
6	5	5	4Ø	1	1.2

Ø
Ø

FINAL EIGENVALUES

	4.55Ø1Ø9524972E+Ø1	2.4271Ø3576410E+Ø1	-7.628352951Ø42E+ØØ	
1 N	NA	NB	KM	EPS
3Ø	5	4	-2ØØ	1.ØØE-1Ø

Ø
Ø
Ø

P	EM(IN)	EM(OUT)	KS	K	MAX RES
2	1	1	3Ø	1	8.4
3	1	1	3	1	5.7
3	2	2	15	1	1.1
4	1	1	3	1	1.2
4	2	2	3	1	9.3
4	3	3	31	1	8.2
5	1	1	3	1	7.3
5	2	2	3	1	5.Ø
5	3	3	3	1	9.1
5	4	4	3	1	6.3
6	1	1	3	1	6.2
6	2	2	3	1	1.3
6	3	3	3	1	8.5
6	4	4	3	1	1.1
6	5	5	18	1	1.3

Ø
Ø

FINAL EIGENVALUES

	4.55Ø1Ø9524972E+Ø1	2.4271Ø3576410E+Ø1	-7.628352951Ø42E+ØØ	
1 N	NA	NB	KM	EPS
4Ø	4	3	2ØØ	1.ØØE-1Ø

Ø
Ø

ADJACENT NON-ZERO LOWER DIAGONALS

5Ø	62	67	76	78	67	67	98	1ØØ
82	66	64	84	66	89	76	5Ø	6Ø
58	61	99	56	1ØØ	91	55	83	68

```
   72   -73   -61    7Ø   -94   -36   -8Ø    29   -13
   -3    76    25   -85    78    21    37     3    77
  -15    52   -96    76   -91    26    52   -39    72

   5Ø    16    35    2Ø    73   -5Ø    76   -75   -27
  -89    95   -96    6Ø    9Ø   -81   -71   -71   -45
  -19    8Ø    4Ø    23   -44    39    35   -91

  -76    97   -47    47    36    74   -44   -18   -71
   24    34   -61   -86    66   -25    1Ø    16    32
   -3    1Ø    55    25   -74   -19    14
Ø

                                ADJACENT NON-ZERO LOWER DIAGONALS

  -53    73   -15   -63    7Ø    39    12   -48   -99
   13   -21    78   -34   -17    62    64   -17   -64
   85   -13     6    12   -2Ø   -76   -2Ø   -57    77

   66    77   -46    66   -49   -86    4Ø   -14    69
    8   -93   -93   -98    28    51    4Ø   -33    53
   73    27   -14    96   -31   -15   -71     6    17

   64     2    49    46   -6Ø    24    63   -5Ø   -22
   54    99    -5    51    -9   -43   -34   -46   -98
   15    -7    -2    33    77    41    72   -81

  -66   -4Ø    42   -45    1Ø   -97   -57   -14   -67
  -48    12    93     Ø    21   -79     7     3   -54
   62    -5    16   -52    82   -28    43

  -77   -79    -4   -63    61    31    -3    55   -24
   83     8   -44    81   -68   -44     9    -8   -68
  -64     8    19   -35    42    99
Ø
Ø
```

P	EM(IN)	EM(OUT)	KS	K	MAX RES
2	1	1	13	1	2.8
3	1	1	13	1	4.2
3	2	2	13	1	4.3
4	1	1	7	1	5.Ø
4	2	2	13	1	4.1
4	3	3	54	1	3.8
5	1	1	7	1	4.9
5	2	2	13	1	5.2
5	3	3	34	1	4.Ø
5	4	4	17Ø	1	5.2
6	1	1	7	1	4.8
6	2	2	13	1	4.4
6	3	3	3Ø	1	4.6
6	4	4	37	1	5.Ø
6	5	5	26	1	6.7
7	1	1	7	1	5.6
7	2	2	13	1	5.6
7	3	3	25	1	5.9
7	4	4	27	1	5.Ø
7	5	5	27	1	5.Ø
7	6	5	91	1	6.6
8	1	1	7	1	5.7
8	2	2	13	1	5.5
8	3	3	3Ø	1	6.9
8	4	4	37	1	6.1
8	5	5	28	1	7.8
8	6	6	18Ø	1	6.6
8	7	6	177	1	7.8

```
Ø
Ø                                                    FINAL EIGENVALUES

                -1.Ø596111Ø8Ø16E+Ø3  -2.114191261132E+Ø1   2.825ØØ2Ø39426E-Ø1
                -6.8Ø939421693ØE-Ø2  -6.2435Ø6342913E-Ø2
```

1 N	NA	NB	KM	EPS
4Ø	4	3	-2ØØ	1.ØØE-1Ø

Ø P	EM(IN)	EM(OUT)	KS	K	MAX RES
2	1	1	13	1	3.1

3	1	1	3	1	3.4
3	2	2	3	1	3.8
4	1	1	3	1	5.6
4	2	2	7	1	4.1
4	3	3	4Ø	1	4.4
5	1	1	3	1	4.Ø
5	2	2	3	1	3.6
5	3	3	18	1	5.2
5	4	4	1ØØ	1	3.8
6	1	1	3	1	6.2
6	2	2	3	1	4.9
6	3	3	11	1	2.8
6	4	4	13	1	5.1
6	5	5	13	1	4.7
7	1	1	3	1	3.9
7	2	2	3	1	4.Ø
7	3	3	7	1	5.7
7	4	4	19	1	3.8
7	5	5	13	1	5.4
7	6	6	187	1	4.2
8	1	1	3	1	4.1
8	2	2	3	1	5.9
8	3	3	7	1	5.5
8	4	4	1Ø	1	4.7
8	5	5	12	1	6.1
8	6	6	55	1	6.1
8	7	6	157	1	4.9

Ø
Ø FINAL EIGENVALUES

-1.Ø596111Ø8Ø16E+Ø3 -2.114191261132E+Ø1 2.825ØØ2Ø39426E-Ø1
-6.8Ø9394215568E-Ø2 -6.2435Ø6344448E-Ø2

1 N	NA	NB	KM	EPS
5Ø	2	1	2ØØ	1.ØØE-1Ø

Ø
Ø ADJACENT NON-ZERO LOWER DIAGONALS

74	79	1ØØ	89	95	93	6Ø	59	57
71	61	74	78	7Ø	95	78	64	76
51	96	98	71	55	72	84	51	66
72	95	62	68	82				
46	11	-81	16	-4Ø	5Ø	-96	12	59
59	9Ø	-54	17	88	-97	-18	79	18
-2Ø	-2Ø	73	2	-16	94	64	86	44
92	-5	1	-63					

Ø
ADJACENT NON-ZERO LOWER DIAGONALS

-4Ø	84	-74	54	-91	8Ø	-9Ø	-6Ø	-92
9Ø	74	-97	-46	-43	-95	37	-12	-89
61	-57	-55	13	44	41	-57	-79	-91
-27	77	-98	62	49				
-67	Ø	-19	68	-12	5Ø	53	-44	-56
-6	-63	-4	-86	-42	2	-26	-3	-5Ø
-38	73	35	-83	32	-58	-32	28	-2Ø
73	26	-66	-18					
13	-74	-67	-2Ø	-77	53	92	-74	71
57	61	56	-91	-62	-9Ø	-14	45	-9Ø
62	3Ø	-37	88	25	-7Ø	-33	-99	26
-34	-47	-78						

Ø

Ø P	EM(IN)	EM(OUT)	KS	K	MAX RES
2	1	1	46	1	7.Ø
3	1	1	43	1	1.8
3	2	2	6Ø	2	2.7
4	1	1	3Ø	1	7.Ø
4	2	2	43	1	9.9
4	3	3	63	3	2.5
5	1	1	26	1	1.1
5	2	2	39	2	2.6
5	3	3	52	1	2.3
5	4	4	61	1	3.Ø

6	1	1	26	1	1.4
6	2	2	39	1	6.5
6	3	3	39	3	5.7
6	4	4	5Ø	2	1.7
6	5	5	1Ø7	1	7.7
7	1	1	24	1	1.2
7	2	2	24	1	1.2
7	3	3	24	3	4.4
7	4	4	33	1	1.3
7	5	5	35	3	3.9
7	6	6	35	3	1.4
8	1	1	24	1	1.4
8	2	2	24	1	1.6
8	3	3	24	1	1.4
8	4	4	24	4	3.9
8	5	5	35	4	2.6
8	6	6	35	4	1.8
8	7	7	114	4	3.2
9	1	1	24	1	1.4
9	2	2	24	1	2.2
9	3	3	24	1	2.3
9	4	4	24	1	2.2
9	5	5	35	4	1.4
9	6	6	24	6	5.Ø
9	7	7	63	5	1.5
9	8	7	63	7	4.2
1Ø	1	1	24	1	1.4
1Ø	2	2	24	1	1.5
1Ø	3	3	24	1	1.8
1Ø	4	4	24	1	1.5
1Ø	5	5	24	1	1.5
1Ø	6	6	24	6	1.7
1Ø	7	7	48	1	2.8
1Ø	8	8	48	8	4.2
1Ø	9	9	48	9	1.Ø

Ø
Ø

FINAL EIGENVALUES

-2.536352449Ø33E-Ø1 2.Ø38265111579E-Ø1 -1.695641Ø85298E-Ø1
-1.194676112328E-Ø1 7.Ø12Ø2388Ø1ØØE-Ø2 6.1ØØØ76826561E-Ø2

1	N	NA	NB	KM	EPS
Ø	5Ø	2	1	-2ØØ	1.ØØE-1Ø

Ø
Ø

P	EM(IN)	EM(OUT)	KS	K	MAX RES
2	1	1	44	1	2.Ø
3	1	1	3	1	1.8
3	2	2	43	2	6.5
4	1	1	3	1	7.9
4	2	2	3	1	1.9
4	3	3	56	3	1.5
5	1	1	3	1	9.6
5	2	2	3	1	1.8
5	3	3	3	1	1.7
5	4	4	44	4	3.5
6	1	1	3	1	1.Ø
6	2	2	3	1	1.8
6	3	3	3	1	1.5
6	4	4	3	1	1.8
6	5	5	1Ø7	5	2.2
7	1	1	3	1	1.Ø
7	2	2	3	1	1.2
7	3	3	3	1	1.7
7	4	4	3	1	1.7
7	5	5	8	1	1.7
7	6	6	6	6	1.6
8	1	1	3	1	1.7
8	2	2	3	1	1.8
8	3	3	3	1	1.5
8	4	4	3	1	2.Ø
8	5	5	4	1	2.1
8	6	6	3	1	1.6
8	7	7	8Ø	7	1.8
9	1	1	3	1	1.5
9	2	2	3	1	1.8
9	3	3	3	1	1.9

9	4	4	3	1	1.4
9	5	5	3	1	1.8
9	6	6	3	1	1.3
9	7	7	3	1	1.7
9	8	8	99	1	1.7
1Ø	1	1	3	1	1.6
1Ø	2	2	3	1	1.4
1Ø	3	3	3	1	1.5
1Ø	4	4	3	1	1.7
1Ø	5	5	3	1	1.6
1Ø	6	6	3	1	1.7
1Ø	7	7	3	1	1.8
1Ø	8	8	18	1	1.8
1Ø	9	9	3	1	1.6

Ø
Ø

FINAL EIGENVALUES

-2.536352449Ø33E-Ø1	2.Ø38265111579E-Ø1	-1.695641Ø85298E-Ø1
-1.194676112328E-Ø1	7.Ø12Ø2388Ø1Ø1E-Ø2	6.1ØØØ76826562E-Ø2

539-P 1- 0

ALGORITHM 539
Basic Linear Algebra Subprograms for Fortran Usage [F1]

C. L. LAWSON
Jet Propulsion Laboratory
R. J. HANSON
Sandia Laboratories, Albuquerque
D. R. KINCAID
The University of Texas, Austin
and
F. T. KROGH
Jet Propulsion Laboratory

Key Words and Phrases: linear algebra, utilities
CR Categories: 4.49, 5.14
Language: Fortran, assembly language

DESCRIPTION

This package complements [1], where further details are given.

REFERENCES

1. LAWSON, C.L., HANSON, R.J., KINCAID, D.R., AND KROGH, F.T. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Software 5*, 3 (September 1979), 308–323.

ALGORITHM

[The contents of this package consists of four files. The first file contains Fortran versions for the BLAS (38 subprograms), programs for testing the BLAS (13 modules), and 18 subprograms from Brent's multiple precision package that are used in the implementation of the extended precision inner products. The remaining 3 files contain Fortran callable assembly language versions for 3 different machines: IBM 360/370 series, CDC 6000 series, and Univac 1100 series. Printed here are the 38 BLAS subprograms. The complete listing is available from the ACM Algorithms Distribution Service.]

Received 13 July 1977.

The work of the first and fourth authors was supported by the National Aeronautics and Space Administration under Contract NAS 7-100. The work of the second author was supported by the U.S. Energy Research and Development Administration (ERDA) under Contract AT (29-1)-789 and (at Washington State University) by the Office of Naval Research under Contract NR 044-457.
Authors' addresses: C.L. Lawson, Jet Propulsion Laboratory, M/S 125-128, 4800 Oak Grove Drive, Pasadena, CA 91103; R.J. Hanson, Numerical Mathematics, Div. 5122, Sandia Laboratories, Albuquerque, NM 87115; D.R. Kincaid, Center for Numerical Analysis, The University of Texas at Austin, Austin, TX 78712; F.T. Krogh, Jet Propulsion Laboratory, M/S 125-128, 4800 Oak Grove Drive, Pasadena, CA 91103.

```
      REAL FUNCTION SDOT(N,SX,INCX,SY,INCY)                              SD1743Ø
C                                                                        SD1744Ø
C     RETURNS THE DOT PRODUCT OF SINGLE PRECISION SX AND SY.             SD1745Ø
C     SDOT = SUM FOR I = Ø TO N-1 OF  SX(LX+I*INCX) * SY(LY+I*INCY),
C     WHERE LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N, AND LY IS
C     DEFINED IN A SIMILAR WAY USING INCY.
C                                                                        SD1746Ø
      REAL SX(1),SY(1)                                                   SD1747Ø
      SDOT = Ø.ØEØ                                                       SD1748Ø
      IF(N.LE.Ø)RETURN                                                   SD1749Ø
      IF(INCX.EQ.INCY) IF(INCX-1)5,2Ø,6Ø                                 SD1750Ø
    5 CONTINUE                                                           SD1751Ø
C                                                                        SD1752Ø
C        CODE FOR UNEQUAL INCREMENTS OR NONPOSITIVE INCREMENTS.          SD1753Ø
C                                                                        SD1754Ø
      IX = 1                                                             SD1755Ø
      IY = 1                                                             SD1756Ø
      IF(INCX.LT.Ø)IX = (-N+1)*INCX + 1                                  SD1757Ø
      IF(INCY.LT.Ø)IY = (-N+1)*INCY + 1                                  SD1758Ø
      DO 1Ø I = 1,N                                                      SD1759Ø
        SDOT = SDOT + SX(IX)*SY(IY)                                      SD1760Ø
        IX = IX + INCX                                                   SD1761Ø
        IY = IY + INCY                                                   SD1762Ø
   1Ø CONTINUE                                                           SD1763Ø
      RETURN                                                             SD1764Ø
C                                                                        SD1765Ø
C        CODE FOR BOTH INCREMENTS EQUAL TO 1                             SD1766Ø
C                                                                        SD1767Ø
C                                                                        SD1768Ø
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 5.    SD1769Ø
C                                                                        SD1770Ø
   2Ø M = MOD(N,5)
      IF( M .EQ. Ø ) GO TO 4Ø                                            SD1772Ø
      DO 3Ø I = 1,M                                                      SD1773Ø
        SDOT = SDOT + SX(I)*SY(I)                                        SD1774Ø
   3Ø CONTINUE                                                           SD1775Ø
      IF( N .LT. 5 ) RETURN                                              SD1776Ø
   4Ø MP1 = M + 1                                                        SD1777Ø
      DO 5Ø I = MP1,N,5                                                  SD1778Ø
        SDOT = SDOT + SX(I)*SY(I) + SX(I + 1)*SY(I + 1) +                SD1779Ø
     $   SX(I + 2)*SY(I + 2) + SX(I + 3)*SY(I + 3) + SX(I + 4)*SY(I + 4) SD1780Ø
   5Ø CONTINUE                                                           SD1781Ø
      RETURN                                                             SD1782Ø
C                                                                        SD1783Ø
C        CODE FOR POSITIVE EQUAL INCREMENTS .NE.1.                       SD1784Ø
C                                                                        SD1785Ø
   6Ø CONTINUE                                                           SD1786Ø
      NS=N*INCX                                                          SD1787Ø
      DO 7Ø I=1,NS,INCX                                                  SD1788Ø
        SDOT = SDOT + SX(I)*SY(I)                                        SD1789Ø
   7Ø   CONTINUE                                                         SD1790Ø
      RETURN                                                             SD1791Ø
      END                                                                SD1792Ø


      DOUBLE PRECISION FUNCTION DSDOT(N,SX,INCX,SY,INCY)                 DS1793Ø
C
C     RETURNS D.P. DOT PRODUCT ACCUMULATED IN D.P., FOR S.P. SX AND SY
C     DSDOT = SUM FOR I = Ø TO N-1 OF  SX(LX+I*INCX) * SY(LY+I*INCY),
C     WHERE LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N, AND LY IS
C     DEFINED IN A SIMILAR WAY USING INCY.
C
      REAL SX(1),SY(1)                                                   DS1797Ø
C                                                                        DS1798Ø
      DSDOT = Ø.DØ                                                       DS1799Ø
      IF(N .LE. Ø)RETURN                                                 DS18ØØØ
      IF(INCX.EQ.INCY.AND.INCX.GT.Ø) GO TO 2Ø                            DS18Ø1Ø
      KX = 1                                                             DS18Ø2Ø
      KY = 1                                                             DS18Ø3Ø
      IF(INCX.LT.Ø) KX = 1+(1-N)*INCX                                    DS18Ø4Ø
      IF(INCY.LT.Ø) KY = 1+(1-N)*INCY                                    DS18Ø5Ø
          DO 1Ø I = 1,N                                                  DS18Ø6Ø
          DSDOT = DSDOT + DBLE(SX(KX))*DBLE(SY(KY))                      DS18Ø7Ø
          KX = KX + INCX                                                 DS18Ø8Ø
          KY = KY + INCY                                                 DS18Ø9Ø
```

```
   1Ø CONTINUE                                                          DS1810Ø
      RETURN                                                            DS1811Ø
   2Ø CONTINUE                                                          DS1812Ø
      NS = N*INCX                                                       DS1813Ø
          DO 3Ø I=1,NS,INCX                                             DS1814Ø
          DSDOT = DSDOT + DBLE(SX(I))*DBLE(SY(I))                       DS1815Ø
   3Ø     CONTINUE                                                      DS1816Ø
      RETURN                                                            DS1817Ø
      END                                                               DS1818Ø


      REAL FUNCTION SDSDOT(N,SB,SX,INCX,SY,INCY)                        SD1819Ø
C
C     RETURNS S.P. RESULT WITH DOT PRODUCT ACCUMULATED IN D.P.
C     SDSDOT = SB + SUM FOR I = Ø TO N-1 OF SX(LX+I*INCX)*SY(LY+I*INCY),
C     WHERE LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N, AND LY IS
C     DEFINED IN A SIMILAR WAY USING INCY.
C
      REAL              SX(1),SY(1),SB                                  SD1824Ø
      DOUBLE PRECISION DSDOT                                            SD1825Ø
C                                                                       SD1826Ø
      DSDOT = DBLE(SB)                                                  SD1827Ø
      IF(N .LE. Ø) GO TO 3Ø                                             SD1828Ø
      IF(INCX.EQ.INCY.AND.INCX.GT.Ø) GO TO 4Ø                           SD1829Ø
      KX = 1                                                            SD183ØØ
      KY = 1                                                            SD1831Ø
      IF(INCX.LT.Ø) KX = 1+(1-N)*INCX                                   SD1832Ø
      IF(INCY.LT.Ø) KY = 1+(1-N)*INCY                                   SD1833Ø
          DO 1Ø I = 1,N                                                 SD1834Ø
          DSDOT = DSDOT + DBLE(SX(KX))*DBLE(SY(KY))                     SD1835Ø
          KX = KX + INCX                                                SD1836Ø
          KY = KY + INCY                                                SD1837Ø
   1Ø     CONTINUE                                                      SD1838Ø
   3Ø SDSDOT = SNGL(DSDOT)                                              SD1839Ø
      RETURN                                                            SD184ØØ
   4Ø CONTINUE                                                          SD1841Ø
      NS = N*INCX                                                       SD1842Ø
          DO 5Ø I=1,NS,INCX                                             SD1843Ø
          DSDOT = DSDOT + DBLE(SX(I))*DBLE(SY(I))                       SD1844Ø
   5Ø     CONTINUE                                                      SD1845Ø
      SDSDOT = SNGL(DSDOT)                                              SD1846Ø
      RETURN                                                            SD1847Ø
      END                                                               SD1848Ø


      DOUBLE PRECISION FUNCTION DDOT(N,DX,INCX,DY,INCY)                 DD1849Ø
C                                                                       DD185ØØ
C     RETURNS THE DOT PRODUCT OF DOUBLE PRECISION DX AND DY.            DD1851Ø
C     DDOT = SUM FOR I = Ø TO N-1 OF  DX(LX+I*INCX) * DY(LY+I*INCY)
C     WHERE LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N, AND LY IS
C     DEFINED IN A SIMILAR WAY USING INCY.
C                                                                       DD1852Ø
      DOUBLE PRECISION DX(1),DY(1)                                      DD1853Ø
      DDOT = Ø.DØ                                                       DD1854Ø
      IF(N.LE.Ø)RETURN                                                  DD1855Ø
      IF(INCX.EQ.INCY) IF(INCX-1) 5,2Ø,6Ø                               DD1856Ø
    5 CONTINUE                                                          DD1857Ø
C                                                                       DD1858Ø
C         CODE FOR UNEQUAL OR NONPOSITIVE INCREMENTS.                   DD1859Ø
C                                                                       DD186ØØ
      IX = 1                                                            DD1861Ø
      IY = 1                                                            DD1862Ø
      IF(INCX.LT.Ø)IX = (-N+1)*INCX + 1                                 DD1863Ø
      IF(INCY.LT.Ø)IY = (-N+1)*INCY + 1                                 DD1864Ø
      DO 1Ø I = 1,N                                                     DD1865Ø
         DDOT = DDOT + DX(IX)*DY(IY)                                    DD1866Ø
        IX = IX + INCX                                                  DD1867Ø
        IY = IY + INCY                                                  DD1868Ø
   1Ø CONTINUE                                                          DD1869Ø
      RETURN                                                            DD187ØØ
C                                                                       DD1871Ø
C        CODE FOR BOTH INCREMENTS EQUAL TO 1.                           DD1872Ø
C                                                                       DD1873Ø
C                                                                       DD1874Ø
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 5.   DD1875Ø
```

```
C                                                                       DD1876Ø
   2Ø M = MOD(N,5)
      IF( M .EQ. Ø ) GO TO 4Ø                                           DD1878Ø
      DO 3Ø I = 1,M                                                     DD1879Ø
         DDOT = DDOT + DX(I)*DY(I)                                      DD1880Ø
   3Ø CONTINUE                                                          DD1881Ø
      IF( N .LT. 5 ) RETURN                                             DD1882Ø
   4Ø MP1 = M + 1                                                       DD1883Ø
      DO 5Ø I = MP1,N,5                                                 DD1884Ø
         DDOT = DDOT + DX(I)*DY(I) + DX(I+1)*DY(I+1) +                  DD1885Ø
     $   DX(I + 2)*DY(I + 2) + DX(I + 3)*DY(I + 3) + DX(I + 4)*DY(I + 4) DD1886Ø
   5Ø CONTINUE                                                          DD1887Ø
      RETURN                                                            DD1888Ø
C                                                                       DD1889Ø
C         CODE FOR POSITIVE EQUAL INCREMENTS .NE.1.                     DD1890Ø
C                                                                       DD1891Ø
   6Ø CONTINUE                                                          DD1892Ø
      NS = N*INCX                                                       DD1893Ø
          DO 7Ø I=1,NC,INCY                                             DD1894Ø
          DDOT = DDOT + DX(I)*DY(I)                                     DD1895Ø
   7Ø     CONTINUE                                                      DD1896Ø
      RETURN                                                            DD1897Ø
      END                                                               DD1898Ø


      DOUBLE PRECISION FUNCTION DQDOTA(N,DB,QC,DX,INCX,DY,INCY)
C     D.P. DOT PRODUCT WITH EXTENDED PRECISION ACCUMULATION (AND RESULT)
C     QC AND DQDOTA ARE SET = DB + QC + SUM FOR I = Ø TO N-1 OF
C       DX(LX+I*INCX) * DY(LY+I*INCY),  WHERE QC IS AN EXTENDED
C       PRECISION RESULT PREVIOUSLY COMPUTED BY DQDOTI OR DQDOTA
C       AND LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N, AND LY IS
C       DEFINED IN A SIMILAR WAY USING INCY. THE MP PACKAGE BY
C       RICHARD P. BRENT IS USED FOR THE EXTENDED PRECISION ARITHMETIC.
C
C     FRED T. KROGH,  JPL,  1977,  JUNE 1
C2
      DOUBLE PRECISION DX(1), DY(1), DB
      INTEGER  QC(1Ø), QX(1Ø), QY(1Ø)
C     THE COMMON BLOCK FOR THE MP PACKAGE (MODIFIED TO GIVE IT A NAME)
      COMMON /MPCOM/  MPB, MPT, MPM, MPLUN, MPMXR, MPR(12)
      DATA  I1 / Ø /
C     IF I1 IS Ø THE MP PACKAGE MUST BE INITIALIZED (MPBLAS SETS I1 = 1)
      IF (I1 .EQ. Ø) CALL MPBLAS(I1)
      IF (DB .EQ. Ø.DØ) GO TO 2Ø
      CALL MPCDM(DB, QX)
      CALL MPADD(QC, QX, QC)
   2Ø IF (N .EQ. Ø) GO TO 4Ø
      IX = 1
      IY = 1
      IF (INCX .LT. Ø) IX = (-N + 1) * INCX + 1
      IF (INCY .LT. Ø) IY = (-N + 1) * INCY + 1
      DO  3Ø  I = 1,N
         CALL MPCDM(DX(IX), QX)
         CALL MPCDM(DY(IY), QY)
         CALL MPMUL(QX, QY, QX)
         CALL MPADD(QC, QX, QC)
         IX = IX + INCX
         IY = IY + INCY
   3Ø CONTINUE
   4Ø CALL MPCMD(QC, DQDOTA)
      RETURN
      END


      COMPLEX FUNCTION CDOTC(N,CX,INCX,CY,INCY)                         CD1978Ø
C
C     RETURNS THE DOT PRODUCT FOR COMPLEX CX AND CY, USES CONJUGATE(CX)
C     CDOTC = SUM FOR I = Ø TO N-1 OF CONJ(CX(LX+I*INCX))*CY(LY+I*INCY),
C     WHERE LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N, AND LY IS
C     DEFINED IN A SIMILAR WAY USING INCY.
C
      COMPLEX CX(1),CY(1)                                               CD1982Ø
C                                                                       CD1983Ø
      CDOTC = (Ø.,Ø.)                                                   CD1984Ø
      IF(N .LE. Ø)RETURN                                                CD1985Ø
      IF(INCX.EQ.INCY.AND.INCX.GT.Ø) GO TO 2Ø                           CD1986Ø
```

```
      KX = 1                                                          CD1987Ø
      KY = 1                                                          CD1988Ø
      IF(INCX.LT.Ø) KX = 1+(1-N)*INCX                                 CD1989Ø
      IF(INCY.LT.Ø) KY = 1+(1-N)*INCY                                 CD199ØØ
          DO 1Ø I = 1,N                                               CD1991Ø
          CDOTC = CDOTC + CONJG(CX(KX))*CY(KY)                        CD1992Ø
          KX = KX + INCX                                              CD1993Ø
          KY = KY + INCY                                              CD1994Ø
   1Ø     CONTINUE                                                    CD1995Ø
      RETURN                                                          CD1996Ø
   2Ø CONTINUE                                                        CD1997Ø
      NS = N*INCX                                                     CD1998Ø
          DO 3Ø I=1,NS,INCX                                           CD1999Ø
          CDOTC = CONJG(CX(I))*CY(I) + CDOTC                          CD2ØØØØ
   3Ø     CONTINUE                                                    CD2ØØ1Ø
      RETURN                                                          CD2ØØ2Ø
      END                                                             CD2ØØ3Ø



      COMPLEX FUNCTION CDOTU(N,CX,INCX,CY,INCY)                       CD2ØØ4Ø
C
C     RETURNS THE DOT PRODUCT FOR COMPLEX CX AND CY, NO CONJUGATION
C     CDOTU = SUM FOR I = Ø TO N-1 OF  CX(LX+I*INCX) * CY(LY+I*INCY),
C     WHERE LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N, AND LY IS
C     DEFINED IN A SIMILAR WAY USING INCY.
C
      COMPLEX CX(1),CY(1)                                             CD2ØØ8Ø
C                                                                     CD2ØØ9Ø
      CDOTU = (Ø.,Ø.)                                                 CD2Ø1ØØ
      IF(N .LE. Ø)RETURN                                              CD2Ø11Ø
      IF(INCX.EQ.INCY.AND.INCX.GT.Ø) GO TO 2Ø                         CD2Ø12Ø
      KX = 1                                                          CD2Ø13Ø
      KY = 1                                                          CD2Ø14Ø
      IF(INCX.LT.Ø) KX = 1+(1-N)*INCX                                 CD2Ø15Ø
      IF(INCY.LT.Ø) KY = 1+(1-N)*INCY                                 CD2Ø16Ø
          DO 1Ø I = 1,N                                               CD2Ø17Ø
          CDOTU = CDOTU + CX(KX)*CY(KY)                               CD2Ø18Ø
          KX = KX + INCX                                              CD2Ø19Ø
          KY = KY + INCY                                              CD2Ø2ØØ
   1Ø     CONTINUE                                                    CD2Ø21Ø
      RETURN                                                          CD2Ø22Ø
   2Ø CONTINUE                                                        CD2Ø23Ø
      NS = N*INCX                                                     CD2Ø24Ø
          DO 3Ø I=1,NS,INCX                                           CD2Ø25Ø
          CDOTU = CDOTU + CX(I)*CY(I)                                 CD2Ø26Ø
   3Ø     CONTINUE                                                    CD2Ø27Ø
      RETURN                                                          CD2Ø28Ø
      END                                                             CD2Ø29Ø



      SUBROUTINE SAXPY(N,SA,SX,INCX,SY,INCY)                          SA2Ø3ØØ
C                                                                     SA2Ø31Ø
C     OVERWRITE SINGLE PRECISION SY WITH SINGLE PRECISION SA*SX +SY.  SA2Ø32Ø
C     FOR I = Ø TO N-1, REPLACE  SY(LY+I*INCY) WITH SA*SX(LX+I*INCX) +
C       SY(LY+I*INCY), WHERE LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N,
C       AND LY IS DEFINED IN A SIMILAR WAY USING INCY.
C                                                                     SA2Ø33Ø
      REAL SX(1),SY(1),SA                                             SA2Ø34Ø
      IF(N.LE.Ø.OR.SA.EQ.Ø.EØ) RETURN                                 SA2Ø35Ø
      IF(INCX.EQ.INCY) IF(INCX-1) 5,2Ø,6Ø                             SA2Ø36Ø
    5 CONTINUE                                                        SA2Ø37Ø
C                                                                     SA2Ø38Ø
C        CODE FOR NONEQUAL OR NONPOSITIVE INCREMENTS.                 SA2Ø39Ø
C                                                                     SA2Ø4ØØ
      IX = 1                                                          SA2Ø41Ø
      IY = 1                                                          SA2Ø42Ø
      IF(INCX.LT.Ø)IX = (-N+1)*INCX + 1                               SA2Ø43Ø
      IF(INCY.LT.Ø)IY = (-N+1)*INCY + 1                               SA2Ø44Ø
      DO 1Ø I = 1,N                                                   SA2Ø45Ø
        SY(IY) = SY(IY) + SA*SX(IX)                                   SA2Ø46Ø
        IX = IX + INCX                                                SA2Ø47Ø
        IY = IY + INCY                                                SA2Ø48Ø
   1Ø CONTINUE                                                        SA2Ø49Ø
      RETURN                                                          SA2Ø5ØØ
```

```
C                                                                       SA2Ø51Ø
C       CODE FOR BOTH INCREMENTS EQUAL TO 1                             SA2Ø52Ø
C                                                                       SA2Ø53Ø
C                                                                       SA2Ø54Ø
C       CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 4.    SA2Ø55Ø
C                                                                       SA2Ø56Ø
   2Ø M = MOD(N,4)
      IF( M .EQ. Ø ) GO TO 4Ø                                           SA2Ø58Ø
      DO 3Ø I = 1,M                                                     SA2Ø59Ø
        SY(I) = SY(I) + SA*SX(I)                                        SA2Ø6ØØ
   3Ø CONTINUE                                                          SA2Ø61Ø
      IF( N .LT. 4 ) RETURN                                             SA2Ø62Ø
   4Ø MP1 = M + 1                                                       SA2Ø63Ø
      DO 5Ø I = MP1,N,4                                                 SA2Ø64Ø
        SY(I) = SY(I) + SA*SX(I)                                        SA2Ø65Ø
        SY(I + 1) = SY(I + 1) + SA*SX(I + 1)                            SA2Ø66Ø
        SY(I + 2) = SY(I + 2) + SA*SX(I + 2)                            SA2Ø67Ø
        SY(I + 3) = SY(I + 3) + SA*SX(I + 3)                            SA2Ø68Ø
   5Ø CONTINUE                                                          SA2Ø69Ø
      RETURN                                                            SA2Ø7ØØ
C                                                                       SA2Ø71Ø
C       CODE FOR EQUAL, POSITIVE, NONUNIT INCREMENTS.                   SA2Ø72Ø
C                                                                       SA2Ø73Ø
   6Ø CONTINUE                                                          SA2Ø74Ø
      NS = N*INCX                                                       SA2Ø75Ø
          DO 7Ø I=1,NS,INCX                                             SA2Ø76Ø
          SY(I) = SA*SX(I) + SY(I)                                      SA2Ø77Ø
   7Ø     CONTINUE                                                     SA2Ø78Ø
      RETURN                                                            SA2Ø79Ø
      END                                                               SA2Ø8ØØ


      SUBROUTINE DAXPY(N,DA,DX,INCX,DY,INCY)                            DA2Ø81Ø
C                                                                       DA2Ø82Ø
C     OVERWRITE DOUBLE PRECISION DY WITH DOUBLE PRECISION DA*DX + DY.   DA2Ø83Ø
C     FOR I = Ø TO N-1, REPLACE  DY(LY+I*INCY) WITH DA*DX(LX+I*INCX) +
C       DY(LY+I*INCY), WHERE LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N,
C       AND LY IS DEFINED IN A SIMILAR WAY USING INCY.
C                                                                       DA2Ø84Ø
      DOUBLE PRECISION DX(1),DY(1),DA                                   DA2Ø85Ø
      IF(N.LE.Ø.OR.DA.EQ.Ø.DØ) RETURN                                   DA2Ø86Ø
      IF(INCX.EQ.INCY) IF(INCX-1) 5,2Ø,6Ø                               DA2Ø87Ø
    5 CONTINUE                                                          DA2Ø88Ø
C                                                                       DA2Ø89Ø
C       CODE FOR NONEQUAL OR NONPOSITIVE INCREMENTS.                    DA2Ø9ØØ
C                                                                       DA2Ø91Ø
      IX = 1                                                            DA2Ø92Ø
      IY = 1                                                            DA2Ø93Ø
      IF(INCX.LT.Ø)IX = (-N+1)*INCX + 1                                 DA2Ø94Ø
      IF(INCY.LT.Ø)IY = (-N+1)*INCY + 1                                 DA2Ø95Ø
      DO 1Ø I = 1,N                                                     DA2Ø96Ø
        DY(IY) = DY(IY) + DA*DX(IX)                                     DA2Ø97Ø
        IX = IX + INCX                                                  DA2Ø98Ø
        IY = IY + INCY                                                  DA2Ø99Ø
   1Ø CONTINUE                                                          DA21ØØØ
      RETURN                                                            DA21Ø1Ø
C                                                                       DA21Ø2Ø
C       CODE FOR BOTH INCREMENTS EQUAL TO 1                             DA21Ø3Ø
C                                                                       DA21Ø4Ø
C                                                                       DA21Ø5Ø
C       CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 4.    DA21Ø6Ø
C                                                                       DA21Ø7Ø
   2Ø M = MOD(N,4)
      IF( M .EQ. Ø ) GO TO 4Ø                                           DA21Ø9Ø
      DO 3Ø I = 1,M                                                     DA211ØØ

        DY(I) = DY(I) + DA*DX(I)                                        DA2111Ø
   3Ø CONTINUE                                                          DA2112Ø
      IF( N .LT. 4 ) RETURN                                             DA2113Ø
   4Ø MP1 = M + 1                                                       DA2114Ø
      DO 5Ø I = MP1,N,4                                                 DA2115Ø
        DY(I) = DY(I) + DA*DX(I)                                        DA2116Ø
        DY(I + 1) = DY(I + 1) + DA*DX(I + 1)                            DA2117Ø
        DY(I + 2) = DY(I + 2) + DA*DX(I + 2)                            DA2118Ø
        DY(I + 3) = DY(I + 3) + DA*DX(I + 3)                            DA2119Ø
   5Ø CONTINUE                                                          DA212ØØ
```

```
      RETURN                                                            DA2121Ø
C                                                                       DA2122Ø
C        CODE FOR EQUAL, POSITIVE, NONUNIT INCREMENTS.                  DA2123Ø
C                                                                       DA2124Ø
   6Ø CONTINUE                                                          DA2125Ø
      NS = N*INCX                                                       DA2126Ø
          DO 7Ø I=1,NS,INCX                                             DA2127Ø
          DY(I) = DA*DX(I) + DY(I)                                      DA2128Ø
   7Ø     CONTINUE                                                      DA2129Ø
      RETURN                                                            DA2130Ø
      END                                                               DA2131Ø


      SUBROUTINE CAXPY(N,CA,CX,INCX,CY,INCY)                            CA2132Ø
C
C     OVERWRITE COMPLEX CY WITH COMPLEX  CA*CX + CY.
C     FOR I = Ø TO N-1, REPLACE  CY(LY+I*INCY) WITH CA*CX(LX+I*INCX) +
C       CY(LY+I*INCY), WHERE LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N,
C       AND LY IS DEFINED IN A SIMILAR WAY USING INCY.
C
      COMPLEX CX(1),CY(1),CA                                            CA2136Ø
C                                                                       CA2137Ø
      CANORM = ABS(REAL(CA)) + ABS(AIMAG(CA))                           CA2138Ø
      IF(N.LE.Ø.OR.CANORM.EQ.Ø.EØ) RETURN                               CA2139Ø
      IF(INCX.EQ.INCY.AND.INCX.GT.Ø) GO TO 2Ø                           CA2140Ø
      KX = 1                                                            CA2141Ø
      KY = 1                                                            CA2142Ø
      IF(INCX.LT.Ø) KX = 1+(1-N)*INCX                                   CA2143Ø
      IF(INCY.LT.Ø) KY = 1+(1-N)*INCY                                   CA2144Ø
          DO 1Ø I = 1,N                                                 CA2145Ø
          CY(KY) = CY(KY) + CA*CX(KX)                                   CA2146Ø
          KX = KX + INCX                                                CA2147Ø
          KY = KY + INCY                                                CA2148Ø
   1Ø CONTINUE                                                          CA2149Ø
      RETURN                                                            CA2150Ø
   2Ø CONTINUE                                                          CA2151Ø
      NS = N*INCX                                                       CA2152Ø
          DO 3Ø I=1,NS,INCX                                             CA2153Ø
          CY(I) = CA*CX(I) + CY(I)                                      CA2154Ø
   3Ø     CONTINUE                                                      CA2155Ø
      RETURN                                                            CA2156Ø
      END                                                               CA2157Ø


      SUBROUTINE SROTG(SA,SB,SC,SS)
C
C     DESIGNED BY C.L.LAWSON, JPL, 1977 SEPT Ø8
C
C
C     CONSTRUCT THE GIVENS TRANSFORMATION
C
C         ( SC  SS )
C     G = (        ) ,    SC**2 + SS**2 = 1 ,
C         (-SS  SC )
C
C     WHICH ZEROS THE SECOND ENTRY OF THE 2-VECTOR  (SA,SB)**T .
C
C     THE QUANTITY R = (+/-)SQRT(SA**2 + SB**2) OVERWRITES SA IN
C     STORAGE.  THE VALUE OF SB IS OVERWRITTEN BY A VALUE Z WHICH
C     ALLOWS SC AND SS TO BE RECOVERED BY THE FOLLOWING ALGORITHM:
C           IF Z=1  SET  SC=Ø.  AND  SS=1.
C           IF ABS(Z) .LT. 1  SET  SC=SQRT(1-Z**2)  AND  SS=Z
C           IF ABS(Z) .GT. 1  SET  SC=1/Z  AND  SS=SQRT(1-SC**2)
C
C     NORMALLY, THE SUBPROGRAM SROT(N,SX,INCX,SY,INCY,SC,SS) WILL
C     NEXT BE CALLED TO APPLY THE TRANSFORMATION TO A 2 BY N MATRIX.
C
C -----------------------------------------------------------------
C
      IF (ABS(SA) .LE. ABS(SB)) GO TO 1Ø
C
C *** HERE ABS(SA) .GT. ABS(SB) ***
C
      U = SA + SA
      V = SB / U
```

```
C
C     NOTE THAT U AND R HAVE THE SIGN OF SA
C
      R = SQRT(.25 + V**2) * U
C
C     NOTE THAT SC IS POSITIVE
C
      SC = SA / R
      SS = V * (SC + SC)
      SB = SS
      SA = R
      RETURN
C
C *** HERE ABS(SA) .LE. ABS(SB) ***
C
   1Ø IF (SB .EQ. Ø.) GO TO 2Ø
      U = SB + SB
      V = SA / U
C

C     NOTE THAT U AND R HAVE THE SIGN OF SB
C     (R IS IMMEDIATELY STORED IN SA)
C
      SA = SQRT(.25 + V**2) * U
C
C     NOTE THAT SS IS POSITIVE
C
      SS = SB / SA
      SC = V * (SS + SS)
      IF (SC .EQ. Ø.) GO TO 15
      SB = 1. / SC
      RETURN
   15 SB = 1.
      RETURN
C
C *** HERE SA = SB = Ø. ***
C
   2Ø SC = 1.
      SS = Ø.
      RETURN
C
      END


      SUBROUTINE DROTG(DA,DB,DC,DS)
C
C     DESIGNED BY C.L.LAWSON, JPL, 1977 SEPT Ø8
C
C
C     CONSTRUCT THE GIVENS TRANSFORMATION
C
C         ( DC  DS )
C     G = (        ) ,    DC**2 + DS**2 = 1 ,
C         (-DS  DC )
C
C     WHICH ZEROS THE SECOND ENTRY OF THE 2-VECTOR  (DA,DB)**T .
C
C     THE QUANTITY R = (+/-)DSQRT(DA**2 + DB**2) OVERWRITES DA IN
C     STORAGE.  THE VALUE OF DB IS OVERWRITTEN BY A VALUE Z WHICH
C     ALLOWS DC AND DS TO BE RECOVERED BY THE FOLLOWING ALGORITHM:
C           IF Z=1  SET  DC=Ø.DØ  AND  DS=1.DØ
C           IF DABS(Z) .LT. 1  SET  DC=DSQRT(1-Z**2)  AND  DS=Z
C           IF DABS(Z) .GT. 1  SET  DC=1/Z  AND  DS=DSQRT(1-DC**2)
C
C     NORMALLY, THE SUBPROGRAM DROT(N,DX,INCX,DY,INCY,DC,DS) WILL
C     NEXT BE CALLED TO APPLY THE TRANSFORMATION TO A 2 BY N MATRIX.
C
C -----------------------------------------------------------------
C
      DOUBLE PRECISION  DA, DB, DC, DS, U, V, R
      IF (DABS(DA) .LE. DABS(DB)) GO TO 1Ø
C

C *** HERE DABS(DA) .GT. DABS(DB) ***
C
```

```
      U = DA + DA
      V = DB / U
C
C     NOTE THAT U AND R HAVE THE SIGN OF DA
C
      R = DSQRT(.25DØ + V**2) * U
C
C     NOTE THAT DC IS POSITIVE
C
      DC = DA / R
      DS = V * (DC + DC)
      DB = DS
      DA = R
      RETURN
C
C *** HERE DABS(DA) .LE. DABS(DB) ***
C
   1Ø IF (DB .EQ. Ø.DØ) GO TO 2Ø
      U = DB + DB
      V = DA / U
C
C     NOTE THAT U AND R HAVE THE SIGN OF DB
C     (R IS IMMEDIATELY STORED IN DA)
C
      DA = DSQRT(.25DØ + V**2) * U
C
C     NOTE THAT DS IS POSITIVE
C
      DS = DB / DA
      DC = V * (DS + DS)
      IF (DC .EQ. Ø.DØ) GO TO 15
      DB = 1.DØ / DC
      RETURN
   15 DB = 1.DØ
      RETURN
C
C *** HERE DA = DB = Ø.DØ ***
C
   2Ø DC = 1.DØ
      DS = Ø.DØ
      RETURN
C
      END


      SUBROUTINE SROT(N,SX,INCX,SY,INCY,SC,SS)                          SR2421Ø
C                                                                       SR2422Ø
C     MULTIPLY THE 2 X 2 MATRIX  ( SC SS) TIMES THE 2 X N MATRIX (SX**T)
C                                (-SS SC)                        (SY**T)
C     WHERE **T INDICATES TRANSPOSE.    THE ELEMENTS OF SX ARE IN

C     SX(LX+I*INCX), I = Ø TO N-1, WHERE LX = 1 IF INCX .GE. Ø, ELSE
C     LX = (-INCX)*N, AND SIMILARLY FOR SY USING LY AND INCY.
      REAL             SX,SY,SC,SS,ZERO,ONE,W,Z
      DIMENSION SX(1),SY(1)                                             SR2426Ø
C                                                                       SR2427Ø
      DATA ZERO,ONE/Ø.EØ,1.EØ/                                          SR2428Ø
      IF(N .LE. Ø .OR. (SS .EQ. ZERO .AND. SC .EQ. ONE)) GO TO 4Ø       SR2429Ø
      IF(.NOT. (INCX .EQ. INCY .AND. INCX .GT. Ø)) GO TO 2Ø             SR243ØØ
C                                                                       SR2431Ø
           NSTEPS=INCX*N                                              SR2432Ø
           DO 1Ø I=1,NSTEPS,INCX                                      SR2433Ø
                W=SX(I)                                               SR2434Ø
                Z=SY(I)                                               SR2435Ø
                SX(I)=SC*W+SS*Z                                       SR2436Ø
                SY(I)=-SS*W+SC*Z                                      SR2437Ø
   1Ø           CONTINUE                                              SR2438Ø
           GO TO 4Ø                                                   SR2439Ø
C                                                                       SR244ØØ
   2Ø CONTINUE                                                        SR2441Ø
           KX=1                                                       SR2444Ø
           KY=1                                                       SR2445Ø
C                                                                       SR2446Ø
           IF(INCX .LT. Ø) KX=1-(N-1)*INCX                            SR2447Ø
           IF(INCY .LT. Ø) KY=1-(N-1)*INCY                            SR2448Ø
```

```
C                                                                       SR2449Ø
          DO 3Ø I=1,N                                                   SR245ØØ
               W=SX(KX)                                                 SR2451Ø
               Z=SY(KY)                                                 SR2452Ø
               SX(KX)=SC*W+SS*Z                                         SR2453Ø
               SY(KY)=-SS*W+SC*Z                                        SR2454Ø
               KX=KX+INCX                                               SR2455Ø
               KY=KY+INCY                                               SR2456Ø
    3Ø         CONTINUE                                                 SR2457Ø
    4Ø CONTINUE                                                         SR2458Ø
C                                                                       SR2459Ø
      RETURN                                                            SR246ØØ
      END                                                               SR2461Ø


      SUBROUTINE DROT(N,DX,INCX,DY,INCY,DC,DS)                          DR2462Ø
C                                                                       DR2463Ø
C     MULTIPLY THE 2 X 2 MATRIX  ( DC DS) TIMES THE 2 X N MATRIX (DX**T)
C                                (-DS DC)                        (DY**T)
C     WHERE **T INDICATES TRANSPOSE.    THE ELEMENTS OF DX ARE IN
C     DX(LX+I*INCX), I = Ø TO N-1, WHERE LX = 1 IF INCX .GE. Ø, ELSE
C     LX = (-INCX)*N, AND SIMILARLY FOR DY USING LY AND INCY.
      DOUBLE PRECISION DX,DY,DC,DS,ZERO,ONE,W,Z                         DR2467Ø
      DIMENSION DX(1),DY(1)                                             DR2468Ø
C                                                                       DR2469Ø
      DATA ZERO,ONE/Ø.DØ,1.DØ/                                          DR247ØØ
      IF(N .LE. Ø .OR. (DS .EQ. ZERO .AND. DC .EQ. ONE)) GO TO 4Ø       DR2471Ø
      IF(.NOT. (INCX .EQ. INCY .AND. INCX .GT. Ø)) GO TO 2Ø             DR2472Ø

C                                                                       DR2473Ø
          NSTEPS=INCX*N                                                 DR2474Ø
          DO 1Ø I=1,NSTEPS,INCX                                         DR2475Ø
               W=DX(I)                                                  DR2476Ø
               Z=DY(I)                                                  DR2477Ø
               DX(I)=DC*W+DS*Z                                          DR2478Ø
               DY(I)=-DS*W+DC*Z                                         DR2479Ø
    1Ø         CONTINUE                                                 DR248ØØ
          GO TO 4Ø                                                      DR2481Ø
C                                                                       DR2482Ø
    2Ø CONTINUE                                                         DR2483Ø
          KX=1                                                          DR2484Ø
          KY=1                                                          DR2485Ø
C                                                                       DR2486Ø
          IF(INCX .LT. Ø) KX=1-(N-1)*INCX                               DR2487Ø
          IF(INCY .LT. Ø) KY=1-(N-1)*INCY                               DR2488Ø
C                                                                       DR2489Ø
          DO 3Ø I=1,N                                                   DR249ØØ
               W=DX(KX)                                                 DR2491Ø
               Z=DY(KY)                                                 DR2492Ø
               DX(KX)=DC*W+DS*Z                                         DR2493Ø
               DY(KY)=-DS*W+DC*Z                                        DR2494Ø
               KX=KX+INCX                                               DR2495Ø
               KY=KY+INCY                                               DR2496Ø
    3Ø         CONTINUE                                                 DR2497Ø
    4Ø CONTINUE                                                         DR2498Ø
C                                                                       DR2499Ø
      RETURN                                                            DR25ØØØ
      END                                                               DR25Ø1Ø


      SUBROUTINE SROTMG (SD1,SD2,SX1,SY1,SPARAM)                        SROØØØ1Ø
C                                                                       SROØØØ2Ø
C     CONSTRUCT THE MODIFIED GIVENS TRANSFORMATION MATRIX H WHICH ZEROS SROØØØ3Ø
C     THE SECOND COMPONENT OF THE 2-VECTOR  (SQRT(SD1)*SX1,SQRT(SD2)*   SROØØØ4Ø
C     SY2)**T.                                                          SROØØØ5Ø
C     WITH SPARAM(1)=SFLAG, H HAS ONE OF THE FOLLOWING FORMS..          SROØØØ6Ø
C                                                                       SROØØØ7Ø
C     SFLAG=-1.EØ     SFLAG=Ø.EØ        SFLAG=1.EØ     SFLAG=-2.EØ      SROØØØ8Ø
C                                                                       SROØØØ9Ø
C       (SH11  SH12)    (1.EØ  SH12)    (SH11  1.EØ)    (1.EØ  Ø.EØ)    SROØØ1ØØ
C     H=(          )    (          )    (          )    (          )    SROØØ11Ø
C       (SH21  SH22),   (SH21  1.EØ),   (-1.EØ SH22),   (Ø.EØ  1.EØ).   SROØØ12Ø
C     LOCATIONS 2-4 OF SPARAM CONTAIN SH11,SH21,SH12, AND SH22          SROØØ13Ø
C     RESPECTIVELY. (VALUES OF 1.EØ, -1.EØ, OR Ø.EØ IMPLIED BY THE     SROØØ14Ø
```

```
C     VALUE OF SPARAM(1) ARE NOT STORED IN SPARAM.)                    SRO00150
C                                                                      SRO00160
C     THE VALUES OF GAMSQ AND RGAMSQ SET IN THE DATA STATEMENT MAY BE  SRO00170
C     INEXACT.  THIS IS OK AS THEY ARE ONLY USED FOR TESTING THE SIZE  SRO00180
C     OF SD1 AND SD2.  ALL ACTUAL SCALING OF DATA IS DONE USING GAM.   SRO00190
C                                                                      SRO00200
      DIMENSION SPARAM(5)                                              SRO00210
C                                                                      SRO00220
      DATA ZERO,ONE,TWO /0.E0,1.E0,2.E0/                               SRO00230
      DATA GAM,GAMSQ,RGAMSQ/4096.E0,1.67772E7,5.96046E-8/              SRO00240
      IF(.NOT. SD1 .LT. ZERO) GO TO 10                                 SRO00250
C       GO ZERO-H-D-AND-SX1..                                          SRO00260
          GO TO 60                                                     SRO00270
   10 CONTINUE                                                         SRO00280
C     CASE-SD1-NONNEGATIVE                                             SRO00290
      SP2=SD2*SY1                                                      SRO00300
      IF(.NOT. SP2 .EQ. ZERO) GO TO 20                                 SRO00310
          SFLAG=-TWO                                                   SRO00320
          GO TO 260                                                    SRO00330
C     REGULAR-CASE..                                                   SRO00340
   20 CONTINUE                                                         SRO00350
      SP1=SD1*SX1                                                      SRO00360
      SQ2=SP2*SY1                                                      SRO00370
      SQ1=SP1*SX1                                                      SRO00380
C                                                                      SRO00390
      IF(.NOT. ABS(SQ1) .GT. ABS(SQ2)) GO TO 40                        SRO00400
          SH21=-SY1/SX1                                                SRO00410
          SH12=SP2/SP1                                                 SRO00420
C                                                                      SRO00430
          SU=ONE-SH12*SH21                                             SRO00440
C                                                                      SRO00450
          IF(.NOT. SU .LE. ZERO) GO TO 30                              SRO00460
C         GO ZERO-H-D-AND-SX1..                                        SRO00470
               GO TO 60                                                SRO00480
   30     CONTINUE                                                     SRO00490
               SFLAG=ZERO                                              SRO00500
               SD1=SD1/SU                                              SRO00510
               SD2=SD2/SU                                              SRO00520
               SX1=SX1*SU                                              SRO00530
C         GO SCALE-CHECK..                                             SRO00540
               GO TO 100                                               SRO00550
   40 CONTINUE                                                         SRO00560
          IF(.NOT. SQ2 .LT. ZERO) GO TO 50                             SRO00570
C         GO ZERO-H-D-AND-SX1..                                        SRO00580
               GO TO 60                                                SRO00590
   50     CONTINUE                                                     SRO00600
               SFLAG=ONE                                               SRO00610
               SH11=SP1/SP2                                            SRO00620
               SH22=SX1/SY1                                            SRO00630
               SU=ONE+SH11*SH22                                        SRO00640
               STEMP=SD2/SU                                            SRO00650
               SD2=SD1/SU                                              SRO00660
               SD1=STEMP                                               SRO00670
               SX1=SY1*SU                                              SRO00680
C         GO SCALE-CHECK                                               SRO00690
               GO TO 100                                               SRO00700
C     PROCEDURE..ZERO-H-D-AND-SX1..                                    SRO00710
   60 CONTINUE                                                         SRO00720
          SFLAG=-ONE                                                   SRO00730
          SH11=ZERO                                                    SRO00740
          SH12=ZERO                                                    SRO00750
          SH21=ZERO                                                    SRO00760
          SH22=ZERO                                                    SRO00770
C                                                                      SRO00780
          SD1=ZERO                                                     SRO00790
          SD2=ZERO                                                     SRO00800
          SX1=ZERO                                                     SRO00810
C         RETURN..                                                     SRO00820
          GO TO 220                                                    SRO00830
C     PROCEDURE..FIX-H..                                               SRO00840
   70 CONTINUE                                                         SRO00850
      IF(.NOT. SFLAG .GE. ZERO) GO TO 90                               SRO00860
C                                                                      SRO00870
          IF(.NOT. SFLAG .EQ. ZERO) GO TO 80                           SRO00880
          SH11=ONE                                                     SRO00890
          SH22=ONE                                                     SRO00900
```

```
            SFLAG=-ONE                                                  SRO00910
            GO TO 90                                                    SRO00920
     80     CONTINUE                                                    SRO00930
            SH21=-ONE                                                   SRO00940
            SH12=ONE                                                    SRO00950
            SFLAG=-ONE                                                  SRO00960
     90 CONTINUE                                                        SRO00970
        GO TO IGO,(120,150,180,210)                                     SRO00980
C       PROCEDURE..SCALE-CHECK                                          SRO00990
    100 CONTINUE                                                        SRO01000
    110     CONTINUE                                                    SRO01010
            IF(.NOT. SD1 .LE. RGAMSQ) GO TO 130                         SRO01020
                 IF(SD1 .EQ. ZERO) GO TO 160                            SRO01030
                 ASSIGN 120 TO IGO                                      SRO01040
C                FIX-H..                                                SRO01050
                 GO TO 70                                               SRO01060
    120          CONTINUE                                               SRO01070
                 SD1=SD1*GAM**2                                         SRO01080
                 SX1=SX1/GAM                                            SRO01090
                 SH11=SH11/GAM                                          SRO01100
                 SH12=SH12/GAM                                          SRO01110
            GO TO 110                                                   SRO01120
    130 CONTINUE                                                        SRO01130
    140     CONTINUE                                                    SRO01140
            IF(.NOT. SD1 .GE. GAMSQ) GO TO 160                          SRO01150
                 ASSIGN 150 TO IGO                                      SRO01160
C                FIX-H..                                                SRO01170
                 GO TO 70                                               SRO01180
    150          CONTINUE                                               SRO01190
                 SD1=SD1/GAM**2                                         SRO01200
                 SX1=SX1*GAM                                            SRO01210
                 SH11=SH11*GAM                                          SRO01220
                 SH12=SH12*GAM                                          SRO01230
            GO TO 140                                                   SRO01240
    160 CONTINUE                                                        SRO01250
    170     CONTINUE                                                    SRO01260
            IF(.NOT. ABS(SD2) .LE. RGAMSQ) GO TO 190                    SRO01270
                 IF(SD2 .EQ. ZERO) GO TO 220                            SRO01280
                 ASSIGN 180 TO IGO                                      SRO01290
C                FIX-H..                                                SRO01300
                 GO TO 70                                               SRO01310
    180          CONTINUE                                               SRO01320
                 SD2=SD2*GAM**2                                         SRO01330
                 SH21=SH21/GAM                                          SRO01340
                 SH22=SH22/GAM                                          SRO01350
            GO TO 170                                                   SRO01360
    190 CONTINUE                                                        SRO01370
    200     CONTINUE                                                    SRO01380
            IF(.NOT. ABS(SD2) .GE. GAMSQ) GO TO 220                     SRO01390
                 ASSIGN 210 TO IGO                                      SRO01400
C                FIX-H..                                                SRO01410
                 GO TO 70                                               SRO01420
    210          CONTINUE                                               SRO01430
                 SD2=SD2/GAM**2                                         SRO01440
                 SH21=SH21*GAM                                          SRO01450
                 SH22=SH22*GAM                                          SRO01460
            GO TO 200                                                   SRO01470
    220 CONTINUE                                                        SRO01480
            IF(SFLAG)250,230,240                                        SRO01490
    230     CONTINUE                                                    SRO01500
                 SPARAM(3)=SH21                                         SRO01510
                 SPARAM(4)=SH12                                         SRO01520
                 GO TO 260                                              SRO01530
    240     CONTINUE                                                    SRO01540
                 SPARAM(2)=SH11                                         SRO01550
                 SPARAM(5)=SH22                                         SRO01560
                 GO TO 260                                              SRO01570
    250     CONTINUE                                                    SRO01580
                 SPARAM(2)=SH11                                         SRO01590
                 SPARAM(3)=SH21                                         SRO01600
                 SPARAM(4)=SH12                                         SRO01610
                 SPARAM(5)=SH22                                         SRO01620
    260 CONTINUE                                                        SRO01630
            SPARAM(1)=SFLAG                                             SRO01640
            RETURN                                                      SRO01650
        END                                                             SRO01660
```

```
      SUBROUTINE DROTMG (DD1,DD2,DX1,DY1,DPARAM)                        DRO0001Ø
C                                                                       DRO0002Ø
C     CONSTRUCT THE MODIFIED GIVENS TRANSFORMATION MATRIX H WHICH ZEROS DRO0003Ø
C     THE SECOND COMPONENT OF THE 2-VECTOR  (DSQRT(DD1)*DX1,DSQRT(DD2)* DRO0004Ø
C     DY2)**T.                                                          DRO0005Ø
C     WITH DPARAM(1)=DFLAG, H HAS ONE OF THE FOLLOWING FORMS..          DRO0006Ø
C                                                                       DRO0007Ø
C     DFLAG=-1.D0     DFLAG=0.D0        DFLAG=1.D0     DFLAG=-2.D0      DRO0008Ø
C                                                                       DRO0009Ø
C       (DH11  DH12)    (1.D0  DH12)    (DH11  1.D0)    (1.D0  0.D0)    DRO0010Ø
C     H=(          )    (          )    (          )    (          )     DRO0011Ø
C       (DH21  DH22),   (DH21  1.D0),   (-1.D0 DH22),   (0.D0  1.D0).   DRO0012Ø
C     LOCATIONS 2-4 OF DPARAM CONTAIN DH11, DH21, DH12, AND DH22        DRO0013Ø
C     RESPECTIVELY. (VALUES OF 1.D0, -1.D0, OR 0.D0 IMPLIED BY THE      DRO0014Ø
C     VALUE OF DPARAM(1) ARE NOT STORED IN DPARAM.)                     DRO0015Ø
C                                                                       DRO0016Ø
C     THE VALUES OF GAMSQ AND RGAMSQ SET IN THE DATA STATEMENT MAY BE   DRO0017Ø
C     INEXACT.  THIS IS OK AS THEY ARE ONLY USED FOR TESTING THE SIZE   DRO0018Ø
C     OF DD1 AND DD2.  ALL ACTUAL SCALING OF DATA IS DONE USING GAM.    DRO0019Ø
C                                                                       DRO0020Ø
      DOUBLE PRECISION GAM,ONE,RGAMSQ,DD2,DH11,DH21,DPARAM,DP2,         DRO0021Ø
     1 DQ2,DU,DY1,ZERO,GAMSQ,DD1,DFLAG,DH12,DH22,DP1,DQ1,               DRO0022Ø
     2 DTEMP,DX1,TWO                                                   DRO0023Ø
      DIMENSION DPARAM(5)                                               DRO0024Ø
C                                                                       DRO0025Ø
      DATA ZERO,ONE,TWO /0.D0,1.D0,2.D0/                                DRO0026Ø
      DATA GAM,GAMSQ,RGAMSQ/4096.D0,16777216.D0,5.9604645D-8/           DRO0027Ø
      IF(.NOT. DD1 .LT. ZERO) GO TO 10                                  DRO0028Ø
C       GO ZERO-H-D-AND-DX1..                                           DRO0029Ø
          GO TO 60                                                      DRO0030Ø
   10 CONTINUE                                                          DRO0031Ø
C     CASE-DD1-NONNEGATIVE                                              DRO0032Ø
      DP2=DD2*DY1                                                       DRO0033Ø
      IF(.NOT. DP2 .EQ. ZERO) GO TO 20                                  DRO0034Ø
          DFLAG=-TWO                                                    DRO0035Ø
          GO TO 260                                                     DRO0036Ø
C     REGULAR-CASE..                                                    DRO0037Ø
   20 CONTINUE                                                          DRO0038Ø
      DP1=DD1*DX1                                                       DRO0039Ø
      DQ2=DP2*DY1                                                       DRO0040Ø
      DQ1=DP1*DX1                                                       DRO0041Ø
C                                                                       DRO0042Ø
      IF(.NOT. DABS(DQ1) .GT. DABS(DQ2)) GO TO 40                       DRO0043Ø
          DH21=-DY1/DX1                                                 DRO0044Ø
          DH12=DP2/DP1                                                  DRO0045Ø
C                                                                       DRO0046Ø
          DU=ONE-DH12*DH21                                              DRO0047Ø
C                                                                       DRO0048Ø
          IF(.NOT. DU .LE. ZERO) GO TO 30                               DRO0049Ø
C         GO ZERO-H-D-AND-DX1..                                         DRO0050Ø
               GO TO 60                                                 DRO0051Ø
   30     CONTINUE                                                      DRO0052Ø
               DFLAG=ZERO                                               DRO0053Ø
               DD1=DD1/DU                                               DRO0054Ø
               DD2=DD2/DU                                               DRO0055Ø
               DX1=DX1*DU                                               DRO0056Ø
C         GO SCALE-CHECK..                                              DRO0057Ø
               GO TO 100                                                DRO0058Ø
   40 CONTINUE                                                          DRO0059Ø
          IF(.NOT. DQ2 .LT. ZERO) GO TO 50                              DRO0060Ø
C         GO ZERO-H-D-AND-DX1..                                         DRO0061Ø
               GO TO 60                                                 DRO0062Ø
   50     CONTINUE                                                      DRO0063Ø
               DFLAG=ONE                                                DRO0064Ø
               DH11=DP1/DP2                                             DRO0065Ø
               DH22=DX1/DY1                                             DRO0066Ø
               DU=ONE+DH11*DH22                                         DRO0067Ø
               DTEMP=DD2/DU                                             DRO0068Ø
               DD2=DD1/DU                                               DRO0069Ø
               DD1=DTEMP                                                DRO0070Ø
               DX1=DY1*DU                                               DRO0071Ø
C         GO SCALE-CHECK                                                DRO0072Ø
               GO TO 100                                                DRO0073Ø
C     PROCEDURE..ZERO-H-D-AND-DX1..                                     DRO0074Ø
   60 CONTINUE                                                          DRO0075Ø
          DFLAG=-ONE                                                    DRO0076Ø
```

```
          DH11=ZERO                                                    DRO00770
          DH12=ZERO                                                    DRO00780
          DH21=ZERO                                                    DRO00790
          DH22=ZERO                                                    DRO00800
C                                                                      DRO00810
          DD1=ZERO                                                     DRO00820
          DD2=ZERO                                                     DRO00830
          DX1=ZERO                                                     DRO00840
C         RETURN..                                                     DRO00850
          GO TO 220                                                    DRO00860
C     PROCEDURE..FIX-H..                                               DRO00870
   70 CONTINUE                                                         DRO00880
      IF(.NOT. DFLAG .GE. ZERO) GO TO 90                               DRO00890
C                                                                      DRO00900
          IF(.NOT. DFLAG .EQ. ZERO) GO TO 80                           DRO00910
          DH11=ONE                                                     DRO00920
          DH22=ONE                                                     DRO00930
          DFLAG=-ONE                                                   DRO00940
          GO TO 90                                                     DRO00950
   80     CONTINUE                                                     DRO00960
          DH21=-ONE                                                    DRO00970
          DH12=ONE                                                     DRO00980
          DFLAG=-ONE                                                   DRO00990
   90 CONTINUE                                                         DRO01000
      GO TO IGO,(120,150,180,210)                                      DRO01010
C     PROCEDURE..SCALE-CHECK                                           DRO01020
  100 CONTINUE                                                         DRO01030
  110     CONTINUE                                                     DRO01040
          IF(.NOT. DD1 .LE. RGAMSQ) GO TO 130                          DRO01050
               IF(DD1 .EQ. ZERO) GO TO 160                             DRO01060
               ASSIGN 120 TO IGO                                       DRO01070
C              FIX-H..                                                 DRO01080
               GO TO 70                                                DRO01090
  120          CONTINUE                                                DRO01100
               DD1=DD1*GAM**2                                          DRO01110
               DX1=DX1/GAM                                             DRO01120
               DH11=DH11/GAM                                           DRO01130
               DH12=DH12/GAM                                           DRO01140
          GO TO 110                                                    DRO01150
  130 CONTINUE                                                         DRO01160
  140     CONTINUE                                                     DRO01170
          IF(.NOT. DD1 .GE. GAMSQ) GO TO 160                           DRO01180
               ASSIGN 150 TO IGO                                       DRO01190
C              FIX-H..                                                 DRO01200
               GO TO 70                                                DRO01210
  150          CONTINUE                                                DRO01220
               DD1=DD1/GAM**2                                          DRO01230
               DX1=DX1*GAM                                             DRO01240
               DH11=DH11*GAM                                           DRO01250
               DH12=DH12*GAM                                           DRO01260
          GO TO 140                                                    DRO01270
  160 CONTINUE                                                         DRO01280
  170     CONTINUE                                                     DRO01290
          IF(.NOT. DABS(DD2) .LE. RGAMSQ) GO TO 190                    DRO01300
               IF(DD2 .EQ. ZERO) GO TO 220                             DRO01310
               ASSIGN 180 TO IGO                                       DRO01320
C              FIX-H..                                                 DRO01330
               GO TO 70                                                DRO01340
  180          CONTINUE                                                DRO01350
               DD2=DD2*GAM**2                                          DRO01360
               DH21=DH21/GAM                                           DRO01370
               DH22=DH22/GAM                                           DRO01380
          GO TO 170                                                    DRO01390
  190 CONTINUE                                                         DRO01400
  200     CONTINUE                                                     DRO01410
          IF(.NOT. DABS(DD2) .GE. GAMSQ) GO TO 220                     DRO01420
               ASSIGN 210 TO IGO                                       DRO01430
C              FIX-H..                                                 DRO01440
               GO TO 70                                                DRO01450
  210          CONTINUE                                                DRO01460
               DD2=DD2/GAM**2                                          DRO01470
               DH21=DH21*GAM                                           DRO01480
               DH22=DH22*GAM                                           DRO01490
          GO TO 200                                                    DRO01500
  220 CONTINUE                                                         DRO01510
          IF(DFLAG)250,230,240                                         DRO01520
```

```
 23Ø      CONTINUE                                                    DROØ153Ø
               DPARAM(3)=DH21                                         DROØ154Ø
               DPARAM(4)=DH12                                         DROØ155Ø
               GO TO 26Ø                                              DROØ156Ø
 24Ø      CONTINUE                                                    DROØ157Ø
               DPARAM(2)=DH11                                         DROØ158Ø
               DPARAM(5)=DH22                                         DROØ159Ø
               GO TO 26Ø                                              DROØ16ØØ
 25Ø      CONTINUE                                                    DROØ161Ø
               DPARAM(2)=DH11                                         DROØ162Ø
               DPARAM(3)=DH21                                         DROØ163Ø
               DPARAM(4)=DH12                                         DROØ164Ø
               DPARAM(5)=DH22                                         DROØ165Ø
 26Ø CONTINUE                                                        DROØ166Ø
          DPARAM(1)=DFLAG                                           DROØ167Ø
          RETURN                                                    DROØ168Ø
      END                                                           DROØ169Ø


      SUBROUTINE SROTM (N,SX,INCX,SY,INCY,SPARAM)                    SR2843Ø
C                                                                    SR2844Ø
C     APPLY THE MODIFIED GIVENS TRANSFORMATION, H, TO THE 2 BY N MATRIX SR2845Ø
C                                                                    SR2846Ø
C     (SX**T) , WHERE **T INDICATES TRANSPOSE. THE ELEMENTS OF SX ARE IN
C     (DX**T)
C
C     SX(LX+I*INCX), I = Ø TO N-1, WHERE LX = 1 IF INCX .GE. Ø, ELSE
C     LX = (-INCX)*N, AND SIMILARLY FOR SY USING USING LY AND INCY.
C     WITH SPARAM(1)=SFLAG, H HAS ONE OF THE FOLLOWING FORMS..       SR2851Ø
C                                                                    SR2852Ø
C     SFLAG=-1.EØ     SFLAG=Ø.EØ        SFLAG=1.EØ     SFLAG=-2.EØ   SR2853Ø
C                                                                    SR2854Ø
C       (SH11  SH12)    (1.EØ  SH12)    (SH11  1.EØ)    (1.EØ  Ø.EØ) SR2855Ø
C     H=(          )    (          )    (          )    (          ) SR2856Ø
C       (SH21  SH22),   (SH21  1.EØ),   (-1.EØ SH22),   (Ø.EØ  1.EØ). SR2857Ø
C     SEE  SROTMG FOR A DESCRIPTION OF DATA STORAGE IN SPARAM.
C                                                                    SR2858Ø
      DIMENSION SX(1),SY(1),SPARAM(5)                                SR2859Ø
      DATA ZERO,TWO/Ø.EØ,2.EØ/                                       SR286ØØ
C                                                                    SR2861Ø

      SFLAG=SPARAM(1)                                                SR2862Ø
      IF(N .LE. Ø .OR.(SFLAG+TWO.EQ.ZERO)) GO TO 14Ø                 SR2863Ø
          IF(.NOT.(INCX.EQ.INCY.AND. INCX .GT.Ø)) GO TO 7Ø           SR2864Ø
C                                                                    SR2865Ø
               NSTEPS=N*INCX                                         SR2866Ø
               IF(SFLAG) 5Ø,1Ø,3Ø                                    SR2867Ø
  1Ø           CONTINUE                                              SR2868Ø
               SH12=SPARAM(4)                                        SR2869Ø
               SH21=SPARAM(3)                                        SR287ØØ
                    DO 2Ø I=1,NSTEPS,INCX                            SR2871Ø
                    W=SX(I)                                          SR2872Ø
                    Z=SY(I)                                          SR2873Ø
                    SX(I)=W+Z*SH12                                   SR2874Ø
                    SY(I)=W*SH21+Z                                   SR2875Ø
  2Ø                CONTINUE                                         SR2876Ø
               GO TO 14Ø                                             SR2877Ø
  3Ø           CONTINUE                                              SR2878Ø
               SH11=SPARAM(2)                                        SR2879Ø
               SH22=SPARAM(5)                                        SR288ØØ
                    DO 4Ø I=1,NSTEPS,INCX                            SR2881Ø
                    W=SX(I)                                          SR2882Ø
                    Z=SY(I)                                          SR2883Ø
                    SX(I)=W*SH11+Z                                   SR2884Ø
                    SY(I)=-W+SH22*Z                                  SR2885Ø
  4Ø                CONTINUE                                         SR2886Ø
               GO TO 14Ø                                             SR2887Ø
  5Ø           CONTINUE                                              SR2888Ø
               SH11=SPARAM(2)                                        SR2889Ø
               SH12=SPARAM(4)                                        SR289ØØ
               SH21=SPARAM(3)                                        SR2891Ø
               SH22=SPARAM(5)                                        SR2892Ø
                    DO 6Ø I=1,NSTEPS,INCX                            SR2893Ø
                    W=SX(I)                                          SR2894Ø
                    Z=SY(I)                                          SR2895Ø
                    SX(I)=W*SH11+Z*SH12                              SR2896Ø
```

```
                    SY(I)=W*SH21+Z*SH22                                   SR2897Ø
   6Ø               CONTINUE                                              SR2898Ø
                GO TO 14Ø                                                 SR2899Ø
   7Ø       CONTINUE                                                      SR29ØØØ
            KX=1                                                          SR29Ø1Ø
            KY=1                                                          SR29Ø2Ø
            IF(INCX .LT. Ø) KX=1+(1-N)*INCX                               SR29Ø3Ø
            IF(INCY .LT. Ø) KY=1+(1-N)*INCY                               SR29Ø4Ø
C                                                                         SR29Ø5Ø
            IF(SFLAG)12Ø,8Ø,1ØØ                                           SR29Ø6Ø
   8Ø       CONTINUE                                                      SR29Ø7Ø
            SH12=SPARAM(4)                                                SR29Ø8Ø
            SH21=SPARAM(3)                                                SR29Ø9Ø
                 DO 9Ø I=1,N                                              SR291ØØ
                 W=SX(KX)                                                 SR2911Ø

                 Z=SY(KY)                                                 SR2912Ø
                 SX(KX)=W+Z*SH12                                          SR2913Ø
                 SY(KY)=W*SH21+Z                                          SR2914Ø
                 KX=KX+INCX                                               SR2915Ø
                 KY=KY+INCY                                               SR2916Ø
   9Ø            CONTINUE                                                 SR2917Ø
            GO TO 14Ø                                                     SR2918Ø
  1ØØ       CONTINUE                                                      SR2919Ø
            SH11=SPARAM(2)                                                SR292ØØ
            SH22=SPARAM(5)                                                SR2921Ø
                 DO 11Ø I=1,N                                             SR2922Ø
                 W=SX(KX)                                                 SR2923Ø
                 Z=SY(KY)                                                 SR2924Ø
                 SX(KX)=W*SH11+Z                                          SR2925Ø
                 SY(KY)=-W+SH22*Z                                         SR2926Ø
                 KX=KX+INCX                                               SR2927Ø
                 KY=KY+INCY                                               SR2928Ø
  11Ø            CONTINUE                                                 SR2929Ø
            GO TO 14Ø                                                     SR293ØØ
  12Ø       CONTINUE                                                      SR2931Ø
            SH11=SPARAM(2)                                                SR2932Ø
            SH12=SPARAM(4)                                                SR2933Ø
            SH21=SPARAM(3)                                                SR2934Ø
            SH22=SPARAM(5)                                                SR2935Ø
                 DO 13Ø I=1,N                                             SR2936Ø
                 W=SX(KX)                                                 SR2937Ø
                 Z=SY(KY)                                                 SR2938Ø
                 SX(KX)=W*SH11+Z*SH12                                     SR2939Ø
                 SY(KY)=W*SH21+Z*SH22                                     SR294ØØ
                 KX=KX+INCX                                               SR2941Ø
                 KY=KY+INCY                                               SR2942Ø
  13Ø            CONTINUE                                                 SR2943Ø
  14Ø       CONTINUE                                                      SR2944Ø
            RETURN                                                        SR2945Ø
            END                                                           SR2946Ø


      SUBROUTINE  SCOPY(N,SX,INCX,SY,INCY)                                SC3Ø53Ø
C                                                                         SC3Ø54Ø
C     COPY SINGLE PRECISION SX TO SINGLE PRECISION SY.                    SC3Ø55Ø
C     FOR I = Ø TO N-1, COPY  SX(LX+I*INCX) TO SY(LY+I*INCY),
C     WHERE LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N, AND LY IS
C     DEFINED IN A SIMILAR WAY USING INCY.
C                                                                         SC3Ø56Ø
      REAL SX(1),SY(1)                                                    SC3Ø57Ø
      IF(N.LE.Ø)RETURN                                                    SC3Ø58Ø
      IF(INCX.EQ.INCY) IF(INCX-1) 5,2Ø,6Ø                                 SC3Ø59Ø
    5 CONTINUE                                                            SC3Ø6ØØ
C                                                                         SC3Ø61Ø
C        CODE FOR UNEQUAL OR NONPOSITIVE INCREMENTS.                      SC3Ø62Ø
C                                                                         SC3Ø63Ø
      IX = 1                                                              SC3Ø64Ø

      IY = 1                                                              SC3Ø65Ø
      IF(INCX.LT.Ø)IX = (-N+1)*INCX + 1                                   SC3Ø66Ø
      IF(INCY.LT.Ø)IY = (-N+1)*INCY + 1                                   SC3Ø67Ø
      DO 1Ø I = 1,N                                                       SC3Ø68Ø
        SY(IY) = SX(IX)                                                   SC3Ø69Ø
        IX = IX + INCX                                                    SC3Ø7ØØ
```

```
      IY = IY + INCY                                                    SC3Ø71Ø
   1Ø CONTINUE                                                          SC3Ø72Ø
      RETURN                                                            SC3Ø73Ø
C                                                                       SC3Ø74Ø
C        CODE FOR BOTH INCREMENTS EQUAL TO 1                            SC3Ø75Ø
C                                                                       SC3Ø76Ø
C                                                                       SC3Ø77Ø
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 7.   SC3Ø78Ø
C                                                                       SC3Ø79Ø
   2Ø M = MOD(N,7)
      IF( M .EQ. Ø ) GO TO 4Ø                                           SC3Ø81Ø
      DO 3Ø I = 1,M                                                     SC3Ø82Ø
        SY(I) = SX(I)                                                   SC3Ø83Ø
   3Ø CONTINUE                                                          SC3Ø84Ø
      IF( N .LT. 7 ) RETURN                                             SC3Ø85Ø
   4Ø MP1 = M + 1                                                       SC3Ø86Ø
      DO 5Ø I = MP1,N,7                                                 SC3Ø87Ø
        SY(I) = SX(I)                                                   SC3Ø88Ø
        SY(I + 1) = SX(I + 1)                                           SC3Ø89Ø
        SY(I + 2) = SX(I + 2)                                           SC3Ø9ØØ
        SY(I + 3) = SX(I + 3)                                           SC3Ø91Ø
        SY(I + 4) = SX(I + 4)                                           SC3Ø92Ø
        SY(I + 5) = SX(I + 5)                                           SC3Ø93Ø
        SY(I + 6) = SX(I + 6)                                           SC3Ø94Ø
   5Ø CONTINUE                                                          SC3Ø95Ø
      RETURN                                                            SC3Ø96Ø
C                                                                       SC3Ø97Ø
C        CODE FOR EQUAL, POSITIVE, NONUNIT INCREMENTS.                  SC3Ø98Ø
C                                                                       SC3Ø99Ø
   6Ø CONTINUE                                                          SC31ØØØ
      NS = N*INCX                                                       SC31Ø1Ø
          DO 7Ø I=1,NS,INCX                                             SC31Ø2Ø
          SY(I) = SX(I)                                                 SC31Ø3Ø
   7Ø     CONTINUE                                                     SC31Ø4Ø
      RETURN                                                            SC31Ø5Ø
      END                                                               SC31Ø6Ø


      SUBROUTINE DCOPY(N,DX,INCX,DY,INCY)                               DC31Ø7Ø
C                                                                       DC31Ø8Ø
C     COPY DOUBLE PRECISION DX TO DOUBLE PRECISION DY.                  DC31Ø9Ø
C     FOR I = Ø TO N-1, COPY DX(LX+I*INCX) TO DY(LY+I*INCY),
C     WHERE LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N, AND LY IS
C     DEFINED IN A SIMILAR WAY USING INCY.
C                                                                       DC311ØØ
      DOUBLE PRECISION DX(1),DY(1)                                      DC3111Ø

      IF(N.LE.Ø)RETURN                                                  DC3112Ø
      IF(INCX.EQ.INCY) IF(INCX-1) 5,2Ø,6Ø                               DC3113Ø
    5 CONTINUE                                                          DC3114Ø
C                                                                       DC3115Ø
C        CODE FOR UNEQUAL OR NONPOSITIVE INCREMENTS.                    DC3116Ø
C                                                                       DC3117Ø
      IX = 1                                                            DC3118Ø
      IY = 1                                                            DC3119Ø
      IF(INCX.LT.Ø)IX = (-N+1)*INCX + 1                                 DC312ØØ
      IF(INCY.LT.Ø)IY = (-N+1)*INCY + 1                                 DC3121Ø
      DO 1Ø I = 1,N                                                     DC3122Ø
        DY(IY) = DX(IX)                                                 DC3123Ø
        IX = IX + INCX                                                  DC3124Ø
        IY = IY + INCY                                                  DC3125Ø
   1Ø CONTINUE                                                          DC3126Ø
      RETURN                                                            DC3127Ø
C                                                                       DC3128Ø
C        CODE FOR BOTH INCREMENTS EQUAL TO 1                            DC3129Ø
C                                                                       DC313ØØ
C                                                                       DC3131Ø
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 7.   DC3132Ø
C                                                                       DC3133Ø
   2Ø M = MOD(N,7)
      IF( M .EQ. Ø ) GO TO 4Ø                                           DC3135Ø
      DO 3Ø I = 1,M                                                     DC3136Ø
        DY(I) = DX(I)                                                   DC3137Ø
   3Ø CONTINUE                                                          DC3138Ø
      IF( N .LT. 7 ) RETURN                                             DC3139Ø
```

```
   4Ø MP1 = M + 1                                                    DC314ØØ
      DO 5Ø I = MP1,N,7                                              DC3141Ø
        DY(I) = DX(I)                                                DC3142Ø
        DY(I + 1) = DX(I + 1)                                        DC3143Ø
        DY(I + 2) = DX(I + 2)                                        DC3144Ø
        DY(I + 3) = DX(I + 3)                                        DC3145Ø
        DY(I + 4) = DX(I + 4)                                        DC3146Ø
        DY(I + 5) = DX(I + 5)                                        DC3147Ø
        DY(I + 6) = DX(I + 6)                                        DC3148Ø
   5Ø CONTINUE                                                       DC3149Ø
      RETURN                                                         DC315ØØ
C                                                                    DC3151Ø
C        CODE FOR EQUAL, POSITIVE, NONUNIT INCREMENTS.               DC3152Ø
C                                                                    DC3153Ø
   6Ø CONTINUE                                                       DC3154Ø
      NS=N*INCX                                                      DC3155Ø
          DO 7Ø I=1,NS,INCX                                          DC3156Ø
          DY(I) = DX(I)                                              DC3157Ø
   7Ø     CONTINUE                                                   DC3158Ø
      RETURN                                                         DC3159Ø
      END                                                            DC316ØØ


      SUBROUTINE CCOPY(N,CX,INCX,CY,INCY)                            CC3161Ø

C
C     COPY COMPLEX CX TO COMPLEX CY.
C     FOR I = Ø TO N-1, COPY CX(LX+I*INCX) TO CY(LY+I*INCY),
C     WHERE LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N, AND LY IS
C     DEFINED IN A SIMILAR WAY USING INCY.
C
      COMPLEX CX(1),CY(1)                                            CC3165Ø
C                                                                    CC3166Ø
      IF(N .LE. Ø)RETURN                                             CC3167Ø
      IF(INCX.EQ.INCY.AND.INCX.GT.Ø) GO TO 2Ø                        CC3168Ø
      KX = 1                                                         CC3169Ø
      KY = 1                                                         CC317ØØ
      IF(INCX.LT.Ø) KX = 1+(1-N)*INCX                                CC3171Ø
      IF(INCY.LT.Ø) KY = 1+(1-N)*INCY                                CC3172Ø
          DO 1Ø I = 1,N                                              CC3173Ø
          CY(KY) = CX(KX)                                            CC3174Ø
          KX = KX + INCX                                             CC3175Ø
          KY = KY + INCY                                             CC3176Ø
   1Ø CONTINUE                                                       CC3177Ø
      RETURN                                                         CC3178Ø
   2Ø CONTINUE                                                       CC3179Ø
      NS = N*INCX                                                    CC318ØØ
          DO 3Ø I=1,NS,INCX                                          CC3181Ø
          CY(I) = CX(I)                                              CC3182Ø
   3Ø     CONTINUE                                                   CC3183Ø
      RETURN                                                         CC3184Ø
      END                                                            CC3185Ø


      SUBROUTINE  SSWAP (N,SX,INCX,SY,INCY)                          SS3186Ø
C                                                                    SS3187Ø
C     INTERCHANGE SINGLE PRECISION SX AND SINGLE PRECISION SY.       SS3188Ø
C     FOR I = Ø TO N-1, INTERCHANGE  SX(LX+I*INCX) AND SY(LY+I*INCY),
C     WHERE LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N, AND LY IS
C     DEFINED IN A SIMILAR WAY USING INCY.
C                                                                    SS3189Ø
      REAL SX(1),SY(1),STEMP1,STEMP2,STEMP3                          SS319ØØ
      IF(N.LE.Ø)RETURN                                               SS3191Ø
      IF(INCX.EQ.INCY) IF(INCX-1) 5,2Ø,6Ø                            SS3192Ø
    5 CONTINUE                                                       SS3193Ø
C                                                                    SS3194Ø
C       CODE FOR UNEQUAL OR NONPOSITIVE INCREMENTS.                  SS3195Ø
C                                                                    SS3196Ø
      IX = 1                                                         SS3197Ø
      IY = 1                                                         SS3198Ø
      IF(INCX.LT.Ø)IX = (-N+1)*INCX + 1                              SS3199Ø
      IF(INCY.LT.Ø)IY = (-N+1)*INCY + 1                              SS32ØØØ
      DO 1Ø I = 1,N                                                  SS32Ø1Ø
        STEMP1 = SX(IX)                                              SS32Ø2Ø
```

```
        SX(IX) = SY(IY)                                                 SS32Ø3Ø
        SY(IY) = STEMP1                                                 SS32Ø4Ø
        IX = IX + INCX                                                  SS32Ø5Ø

        IY = IY + INCY                                                  SS32Ø6Ø
   1Ø CONTINUE                                                          SS32Ø7Ø
      RETURN                                                            SS32Ø8Ø
C                                                                       SS32Ø9Ø
C       CODE FOR BOTH INCREMENTS EQUAL TO 1                             SS321ØØ
C                                                                       SS3211Ø
C                                                                       SS3212Ø
C       CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 3.    SS3213Ø
C                                                                       SS3214Ø
   2Ø M = MOD(N,3)
      IF( M .EQ. Ø ) GO TO 4Ø                                           SS3216Ø
      DO 3Ø I = 1,M                                                     SS3217Ø
        STEMP1 = SX(I)                                                  SS3218Ø
        SX(I) = SY(I)                                                   SS3219Ø
        SY(I) = STEMP1                                                  SS322ØØ
   3Ø CONTINUE                                                          SS3221Ø
      IF( N .LT. 3 ) RETURN                                             SS3222Ø
   4Ø MP1 = M + 1                                                       SS3223Ø
      DO 5Ø I = MP1,N,3                                                 SS3224Ø
        STEMP1 = SX(I)                                                  SS3225Ø
        STEMP2 = SX(I+1)                                                SS3226Ø
        STEMP3 = SX(I+2)                                                SS3227Ø
        SX(I) = SY(I)                                                   SS3228Ø
        SX(I+1) = SY(I+1)                                               SS3229Ø
        SX(I+2) = SY(I+2)                                               SS323ØØ
        SY(I) = STEMP1                                                  SS3231Ø
        SY(I+1) = STEMP2                                                SS3232Ø
        SY(I+2) = STEMP3                                                SS3233Ø
   5Ø CONTINUE                                                          SS3234Ø
      RETURN                                                            SS3235Ø
   6Ø CONTINUE                                                          SS3236Ø
C                                                                       SS3237Ø
C     CODE FOR EQUAL, POSITIVE, NONUNIT INCREMENTS.                     SS3238Ø
C                                                                       SS3239Ø
      NS = N*INCX                                                       SS324ØØ
        DO 7Ø I=1,NS,INCX                                               SS3241Ø
        STEMP1 = SX(I)                                                  SS3242Ø
        SX(I) = SY(I)                                                   SS3243Ø
        SY(I) = STEMP1                                                  SS3244Ø
   7Ø   CONTINUE                                                        SS3245Ø
      RETURN                                                            SS3246Ø
      END                                                               SS3247Ø


      SUBROUTINE DSWAP(N,DX,INCX,DY,INCY)                               DS3248Ø
C                                                                       DS3249Ø
C     INTERCHANGE DOUBLE PRECISION DX AND DOUBLE PRECISION DY.          DS325ØØ
C     FOR I = Ø TO N-1, INTERCHANGE  DX(LX+I*INCX) AND DY(LY+I*INCY),
C     WHERE LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N, AND LY IS
C     DEFINED IN A SIMILAR WAY USING INCY.
C                                                                       DS3251Ø
      DOUBLE PRECISION DX(1),DY(1),DTEMP1,DTEMP2,DTEMP3                 DS3252Ø

      IF(N.LE.Ø)RETURN                                                  DS3253Ø
      IF(INCX.EQ.INCY) IF(INCX-1) 5,2Ø,6Ø                               DS3254Ø
    5 CONTINUE                                                          DS3255Ø
C                                                                       DS3256Ø
C       CODE FOR UNEQUAL OR NONPOSITIVE INCREMENTS.                     DS3257Ø
C                                                                       DS3258Ø
      IX = 1                                                            DS3259Ø
      IY = 1                                                            DS326ØØ
      IF(INCX.LT.Ø)IX = (-N+1)*INCX + 1                                 DS3261Ø
      IF(INCY.LT.Ø)IY = (-N+1)*INCY + 1                                 DS3262Ø
      DO 1Ø I = 1,N                                                     DS3263Ø
        DTEMP1 = DX(IX)                                                 DS3264Ø
        DX(IX) = DY(IY)                                                 DS3265Ø
        DY(IY) = DTEMP1                                                 DS3266Ø
        IX = IX + INCX                                                  DS3267Ø
        IY = IY + INCY                                                  DS3268Ø
   1Ø CONTINUE                                                          DS3269Ø
      RETURN                                                            DS327ØØ
C                                                                       DS3271Ø
```

```
C       CODE FOR BOTH INCREMENTS EQUAL TO 1                              DS3272Ø
C                                                                        DS3273Ø
C                                                                        DS3274Ø
C       CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 3.     DS3275Ø
C                                                                        DS3276Ø
   2Ø M = MOD(N,3)
      IF( M .EQ. Ø ) GO TO 4Ø                                            DS3278Ø
      DO 3Ø I = 1,M                                                      DS3279Ø
        DTEMP1 = DX(I)                                                   DS328ØØ
        DX(I) = DY(I)                                                    DS3281Ø
        DY(I) = DTEMP1                                                   DS3282Ø
   3Ø CONTINUE                                                           DS3283Ø
      IF( N .LT. 3 ) RETURN                                              DS3284Ø
   4Ø MP1 = M + 1                                                        DS3285Ø
      DO 5Ø I = MP1,N,3                                                  DS3286Ø
        DTEMP1 = DX(I)                                                   DS3287Ø
        DTEMP2 = DX(I+1)                                                 DS3288Ø
        DTEMP3 = DX(I+2)                                                 DS3289Ø
        DX(I) = DY(I)                                                    DS329ØØ
        DX(I+1) = DY(I+1)                                                DS3291Ø
        DX(I+2) = DY(I+2)                                                DS3292Ø
        DY(I) = DTEMP1                                                   DS3293Ø
        DY(I+1) = DTEMP2                                                 DS3294Ø
        DY(I+2) = DTEMP3                                                 DS3295Ø
   5Ø CONTINUE                                                           DS3296Ø
      RETURN                                                             DS3297Ø
   6Ø CONTINUE                                                           DS3298Ø
C                                                                        DS3299Ø
C     CODE FOR EQUAL, POSITIVE, NONUNIT INCREMENTS.                      DS33ØØØ
C                                                                        DS33Ø1Ø
      NS = N*INCX                                                        DS33Ø2Ø
        DO 7Ø I=1,NS,INCX                                                DS33Ø3Ø
        DTEMP1 = DX(I)                                                   DS33Ø4Ø
        DX(I) = DY(I)                                                    DS33Ø5Ø
        DY(I) = DTEMP1                                                   DS33Ø6Ø
   7Ø   CONTINUE                                                         DS33Ø7Ø
      RETURN                                                             DS33Ø8Ø
      END                                                                DS33Ø9Ø


      SUBROUTINE CSWAP(N,CX,INCX,CY,INCY)                                CS331ØØ
C
C     INTERCHANGE COMPLEX CX AND COMPLEX CY
C     FOR I = Ø TO N-1, INTERCHANGE  CX(LX+I*INCX) AND CY(LY+I*INCY),
C     WHERE LX = 1 IF INCX .GT. Ø, ELSE LX = (-INCX)*N, AND LY IS
C     DEFINED IN A SIMILAR WAY USING INCY.
C
      COMPLEX CX(1),CY(1),CTEMP                                          CS3314Ø
C                                                                        CS3315Ø
      IF(N .LE. Ø)RETURN                                                 CS3316Ø
      IF(INCX.EQ.INCY.AND.INCX.GT.Ø) GO TO 2Ø                            CS3317Ø
      KX = 1                                                             CS3318Ø
      KY = 1                                                             CS3319Ø
      IF(INCX.LT.Ø) KX = 1+(1-N)*INCX                                    CS332ØØ
      IF(INCY.LT.Ø) KY = 1+(1-N)*INCY                                    CS3321Ø
          DO 1Ø I = 1,N                                                  CS3322Ø
          CTEMP = CX(KX)                                                 CS3323Ø
          CX(KX) = CY(KY)                                                CS3324Ø
          CY(KY) = CTEMP                                                 CS3325Ø
          KX = KX + INCX                                                 CS3326Ø
          KY = KY + INCY                                                 CS3327Ø
   1Ø CONTINUE                                                           CS3328Ø
      RETURN                                                             CS3329Ø
   2Ø CONTINUE                                                           CS333ØØ
      NS = N*INCX                                                        CS3331Ø
          DO 3Ø I=1,NS,INCX                                              CS3332Ø
          CTEMP = CX(I)                                                  CS3333Ø
          CX(I) = CY(I)                                                  CS3334Ø
          CY(I) = CTEMP                                                  CS3335Ø
   3Ø     CONTINUE                                                       CS3336Ø
      RETURN                                                             CS3337Ø
      END                                                                CS3338Ø
```

```
      REAL FUNCTION SNRM2 ( N, SX, INCX)
      INTEGER          NEXT
      REAL   SX(1),  CUTLO, CUTHI, HITEST, SUM, XMAX, ZERO, ONE
      DATA   ZERO, ONE /Ø.ØEØ, 1.ØEØ/
C
C     EUCLIDEAN NORM OF THE N-VECTOR STORED IN SX() WITH STORAGE
C     INCREMENT INCX .
C     IF    N .LE. Ø RETURN WITH RESULT = Ø.
C     IF N .GE. 1 THEN INCX MUST BE .GE. 1
C
C           C.L.LAWSON, 1978 JAN Ø8

C
C     FOUR PHASE METHOD     USING TWO BUILT-IN CONSTANTS THAT ARE
C     HOPEFULLY APPLICABLE TO ALL MACHINES.
C         CUTLO = MAXIMUM OF  SQRT(U/EPS)  OVER ALL KNOWN MACHINES.
C         CUTHI = MINIMUM OF  SQRT(V)      OVER ALL KNOWN MACHINES.
C     WHERE
C         EPS = SMALLEST NO. SUCH THAT EPS + 1. .GT. 1.
C         U   = SMALLEST POSITIVE NO.   (UNDERFLOW LIMIT)
C         V   = LARGEST  NO.            (OVERFLOW  LIMIT)
C
C     BRIEF OUTLINE OF ALGORITHM..
C
C     PHASE 1    SCANS ZERO COMPONENTS.
C     MOVE TO PHASE 2 WHEN A COMPONENT IS NONZERO AND .LE. CUTLO
C     MOVE TO PHASE 3 WHEN A COMPONENT IS .GT. CUTLO
C     MOVE TO PHASE 4 WHEN A COMPONENT IS .GE. CUTHI/M
C     WHERE M = N FOR X() REAL AND M = 2*N FOR COMPLEX.
C
C     VALUES FOR CUTLO AND CUTHI..
C     FROM THE ENVIRONMENTAL PARAMETERS LISTED IN THE IMSL CONVERTER
C     DOCUMENT THE LIMITING VALUES ARE AS FOLLOWS..
C     CUTLO, S.P.   U/EPS = 2**(-1Ø2) FOR  HONEYWELL.  CLOSE SECONDS ARE
C                   UNIVAC AND DEC AT 2**(-1Ø3)
C                   THUS CUTLO = 2**(-51) = 4.44Ø89E-16
C     CUTHI, S.P.   V = 2**127 FOR UNIVAC, HONEYWELL, AND DEC.
C                   THUS CUTHI = 2**(63.5) = 1.3Ø438E19
C     CUTLO, D.P.   U/EPS = 2**(-67) FOR HONEYWELL AND DEC.
C                   THUS CUTLO = 2**(-33.5) = 8.23181D-11
C     CUTHI, D.P.   SAME AS S.P.  CUTHI = 1.3Ø438D19
C     DATA CUTLO, CUTHI / 8.232D-11,  1.3Ø4D19 /
C     DATA CUTLO, CUTHI / 4.441E-16,  1.3Ø4E19 /
      DATA CUTLO, CUTHI / 4.441E-16,  1.3Ø4E19 /
C
      IF(N .GT. Ø) GO TO 1Ø
         SNRM2  = ZERO
         GO TO 3ØØ
C
   1Ø ASSIGN 3Ø TO NEXT
      SUM = ZERO
      NN = N * INCX
C                                                 BEGIN MAIN LOOP
      I = 1
   2Ø    GO TO NEXT,(3Ø, 5Ø, 7Ø, 11Ø)
   3Ø IF( ABS(SX(I)) .GT. CUTLO) GO TO 85
      ASSIGN 5Ø TO NEXT
      XMAX = ZERO
C
C                       PHASE 1.  SUM IS ZERO
C
   5Ø IF( SX(I) .EQ. ZERO) GO TO 2ØØ
      DNRM2 = XMAX * DSQRT(SUM)
  3ØØ CONTINUE
      RETURN
      END



      REAL FUNCTION SCNRM2( N, CX, INCX)
      LOGICAL IMAG, SCALE
      INTEGER          NEXT
      REAL         CUTLO, CUTHI, HITEST, SUM, XMAX, ABSX, ZERO, ONE
      COMPLEX      CX(1)
      DATA         ZERO, ONE /Ø.ØEØ, 1.ØEØ/
C
```

```
C     UNITARY NORM OF THE COMPLEX N-VECTOR STORED IN CX() WITH STORAGE
C     INCREMENT INCX .
C     IF    N .LE. Ø RETURN WITH RESULT = Ø.
C     IF N .GE. 1 THEN INCX MUST BE .GE. 1
C
C          C.L.LAWSON , 1978 JAN Ø8
C
C     FOUR PHASE METHOD     USING TWO BUILT-IN CONSTANTS THAT ARE
C     HOPEFULLY APPLICABLE TO ALL MACHINES.
C         CUTLO = MAXIMUM OF  SQRT(U/EPS)  OVER ALL KNOWN MACHINES.
C         CUTHI = MINIMUM OF  SQRT(V)      OVER ALL KNOWN MACHINES.
C     WHERE
C         EPS = SMALLEST NO. SUCH THAT EPS + 1. .GT. 1.
C         U   = SMALLEST POSITIVE NO.   (UNDERFLOW LIMIT)
C         V   = LARGEST  NO.            (OVERFLOW  LIMIT)
C
C     BRIEF OUTLINE OF ALGORITHM..
C
C     PHASE 1    SCANS ZERO COMPONENTS.
C     MOVE TO PHASE 2 WHEN A COMPONENT IS NONZERO AND .LE. CUTLO
C     MOVE TO PHASE 3 WHEN A COMPONENT IS .GT. CUTLO
C     MOVE TO PHASE 4 WHEN A COMPONENT IS .GE. CUTHI/M
C     WHERE M = N FOR X() REAL AND M = 2*N FOR COMPLEX.
C
C     VALUES FOR CUTLO AND CUTHI..
C     FROM THE ENVIRONMENTAL PARAMETERS LISTED IN THE IMSL CONVERTER
C     DOCUMENT THE LIMITING VALUES ARE AS FOLLOWS..
C     CUTLO, S.P.   U/EPS = 2**(-1Ø2) FOR  HONEYWELL.  CLOSE SECONDS ARE
C                   UNIVAC AND DEC AT 2**(-1Ø3)
C                   THUS CUTLO = 2**(-51) = 4.44Ø89E-16
C     CUTHI, S.P.   V = 2**127 FOR UNIVAC, HONEYWELL, AND DEC.
C                   THUS CUTHI = 2**(63.5) = 1.3Ø438E19
C     CUTLO, D.P.   U/EPS = 2**(-67) FOR HONEYWELL AND DEC.
C                   THUS CUTLO = 2**(-33.5) = 8.23181D-11
C     CUTHI, D.P.   SAME AS S.P.  CUTHI = 1.3Ø438D19
C     DATA CUTLO, CUTHI / 8.232D-11,  1.3Ø4D19 /
C     DATA CUTLO, CUTHI / 4.441E-16,  1.3Ø4E19 /
      DATA CUTLO, CUTHI / 4.441E-16,  1.3Ø4E19 /
C
      IF(N .GT. Ø) GO TO 1Ø
         SCNRM2  = ZERO
         GO TO 3ØØ
C
   1Ø ASSIGN 3Ø TO NEXT
      SUM = ZERO
      NN = N * INCX
C                                                 BEGIN MAIN LOOP
      DO 21Ø I=1,NN,INCX
         ABSX = ABS(REAL(CX(I)))
         IMAG = .FALSE.
         GO TO NEXT,(3Ø, 5Ø, 7Ø, 9Ø, 11Ø)
   3Ø IF( ABSX .GT. CUTLO) GO TO 85
      ASSIGN 5Ø TO NEXT
      SCALE = .FALSE.
C
C                       PHASE 1.  SUM IS ZERO
C
   5Ø IF( ABSX .EQ. ZERO) GO TO 2ØØ
      IF( ABSX .GT. CUTLO) GO TO 85
C
C                               PREPARE FOR PHASE 2.
      ASSIGN 7Ø TO NEXT
      GO TO 1Ø5
C
C                               PREPARE FOR PHASE 4.
C
  1ØØ ASSIGN 11Ø TO NEXT
      SUM = (SUM / ABSX) / ABSX
  1Ø5 SCALE = .TRUE.
      XMAX = ABSX
      GO TO 115
C
C                  PHASE 2.  SUM IS SMALL.
C                            SCALE TO AVOID DESTRUCTIVE UNDERFLOW.
C
   7Ø IF( ABSX .GT. CUTLO ) GO TO 75
```

```
C
C                    COMMON CODE FOR PHASES 2 AND 4.
C                    IN PHASE 4 SUM IS LARGE.  SCALE TO AVOID OVERFLOW.
C
  11Ø IF( ABSX .LE. XMAX ) GO TO 115
         SUM = ONE + SUM * (XMAX / ABSX)**2
         XMAX = ABSX
         GO TO 2ØØ
C
  115 SUM = SUM + (ABSX/XMAX)**2
      GO TO 2ØØ
C
C
C                  PREPARE FOR PHASE 3.
C
   75 SUM = (SUM * XMAX) * XMAX
C
   85 ASSIGN 9Ø TO NEXT
      SCALE = .FALSE.
C
C     FOR REAL OR D.P. SET HITEST = CUTHI/N
C     FOR COMPLEX      SET HITEST = CUTHI/(2*N)
C
      HITEST = CUTHI/FLOAT( N )
C
C                   PHASE 3.  SUM IS MID-RANGE.  NO SCALING.
C
   9Ø IF(ABSX .GE. HITEST) GO TO 1ØØ
         SUM = SUM + ABSX**2
  2ØØ CONTINUE
C                   CONTROL SELECTION OF REAL AND IMAGINARY PARTS.
C
      IF(IMAG) GO TO 21Ø
         ABSX = ABS(AIMAG(CX(I)))
         IMAG = .TRUE.
      GO TO NEXT,(  5Ø, 7Ø, 9Ø, 11Ø )
C
  21Ø CONTINUE
         IMAG = .TRUE.
      GO TO NEXT,(  5Ø, 7Ø, 9Ø, 11Ø )
C
  21Ø CONTINUE
C
C             END OF MAIN LOOP.
C             COMPUTE SQUARE ROOT AND ADJUST FOR SCALING.
C
      SCNRM2 = SQRT(SUM)
      IF(SCALE) SCNRM2 = SCNRM2 * XMAX
  3ØØ CONTINUE
      RETURN
      END


      REAL FUNCTION SASUM(N,SX,INCX)                                    SA3581Ø
C                                                                       SA3582Ø
C     RETURNS SUM OF MAGNITUDES OF SINGLE PRECISION SX.                 SA3583Ø
C     SASUM = SUM FROM Ø TO N-1 OF  ABS(SX(1+I*INCX))
C                                                                       SA3584Ø
      REAL SX(1)                                                        SA3585Ø
      SASUM = Ø.ØEØ                                                     SA3586Ø
      IF(N.LE.Ø)RETURN                                                  SA3587Ø
      IF(INCX.EQ.1)GOTO 2Ø                                              SA3588Ø
C                                                                       SA3589Ø
C        CODE FOR INCREMENTS NOT EQUAL TO 1.                            SA359ØØ
C                                                                       SA3591Ø
      NS = N*INCX                                                       SA3592Ø
          DO 1Ø I=1,NS,INCX                                             SA3593Ø
          SASUM = SASUM + ABS(SX(I))                                    SA3594Ø
   1Ø     CONTINUE                                                      SA3595Ø
      RETURN                                                            SA3596Ø
C                                                                       SA3597Ø
C        CODE FOR INCREMENTS EQUAL TO 1.                                SA3598Ø
C                                                                       SA3599Ø
C                                                                       SA36ØØØ
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 6.   SA361Ø
C                                                                       SA362Ø
```

```
   2Ø M = MOD(N,6)
      IF( M .EQ. Ø ) GO TO 4Ø                                          SA36Ø4Ø
      DO 3Ø I = 1,M                                                    SA36Ø5Ø
        SASUM = SASUM + ABS(SX(I))                                     SA36Ø6Ø
   3Ø CONTINUE                                                         SA36Ø7Ø
      IF( N .LT. 6 ) RETURN                                            SA36Ø8Ø
   4Ø MP1 = M + 1                                                      SA36Ø9Ø
      DO 5Ø I = MP1,N,6                                                SA361Ø0
        SASUM = SASUM + ABS(SX(I)) + ABS(SX(I + 1)) + ABS(SX(I + 2))   SA361Ø1Ø
     $  + ABS(SX(I + 3)) + ABS(SX(I + 4)) + ABS(SX(I + 5))             SA361Ø2Ø
   5Ø CONTINUE                                                         SA361Ø3Ø
      RETURN                                                           SA361Ø4Ø
      END                                                              SA361Ø5Ø


      DOUBLE PRECISION FUNCTION DASUM(N,DX,INCX)                       DA361Ø6Ø
C                                                                      DA361Ø7Ø
C     RETURNS SUM OF MAGNITUDES OF DOUBLE PRECISION DX.                DA361Ø8Ø
C     DASUM = SUM FROM Ø TO N-1 OF DABS(DX(1+I*INCX))
C                                                                      DA361Ø9Ø
      DOUBLE PRECISION DX(1)                                           DA362ØØ
      DASUM = Ø.DØ                                                     DA3621Ø
      IF(N.LE.Ø)RETURN                                                 DA3622Ø
      IF(INCX.EQ.1)GOTO 2Ø                                             DA3623Ø
C                                                                      DA3624Ø
C        CODE FOR INCREMENTS NOT EQUAL TO 1.                           DA3625Ø
C                                                                      DA3626Ø
      NS = N*INCX                                                      DA3627Ø
          DO 1Ø I=1,NS,INCX                                            DA3628Ø
          DASUM = DASUM + DABS(DX(I))                                  DA3629Ø
   1Ø     CONTINUE                                                     DA363ØØ
      RETURN                                                           DA3631Ø
C                                                                      DA3632Ø
C        CODE FOR INCREMENTS EQUAL TO 1.                               DA3633Ø
C                                                                      DA3634Ø
C                                                                      DA3635Ø
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 6.  DA3636Ø
C                                                                      DA3637Ø
   2Ø M = MOD(N,6)
      IF( M .EQ. Ø ) GO TO 4Ø                                          DA3639Ø
      DO 3Ø I = 1,M                                                    DA364ØØ
        DASUM = DASUM + DABS(DX(I))                                    DA3641Ø
   3Ø CONTINUE                                                         DA3642Ø
      IF( N .LT. 6 ) RETURN                                            DA3643Ø
   4Ø MP1 = M + 1                                                      DA3644Ø
      DO 5Ø I = MP1,N,6                                                DA3645Ø
        DASUM = DASUM + DABS(DX(I)) + DABS(DX(I+1)) + DABS(DX(I+2))    DA3646Ø
     $  + DABS(DX(I+3)) + DABS(DX(I+4)) + DABS(DX(I+5))                DA3647Ø
   5Ø CONTINUE                                                         DA3648Ø
      RETURN                                                           DA3649Ø
      END                                                              DA365ØØ


      FUNCTION SCASUM(N,CX,INCX)                                       SC3651Ø
C     RETURNS SUMS OF MAGNITUDES OF REAL AND IMAGINARY PARTS OF
C     COMPONENTS OF CX.  NOTE THAT THIS IS NOT THE L1 NORM OF CX.
C     CASUM = SUM FROM Ø TO N-1 OF ABS(REAL(CX(1+I*INCX))) +
C             ABS(IMAG(CX(1+I*INCX)))
C
      COMPLEX CX(1)                                                    SC3656Ø
C                                                                      SC3657Ø
      SCASUM=Ø.                                                        SC3658Ø
      IF(N .LE. Ø) RETURN                                              SC3659Ø
      NS = N*INCX                                                      SC366ØØ
          DO 1Ø I=1,NS,INCX                                            SC3661Ø
          SCASUM = SCASUM + ABS(REAL(CX(I))) + ABS(AIMAG(CX(I)))       SC3662Ø
   1Ø     CONTINUE                                                     SC3663Ø
      RETURN                                                           SC3664Ø
      END                                                              SC3665Ø


      SUBROUTINE  SSCAL(N,SA,SX,INCX)                                  SS3666Ø
C                                                                      SS3667Ø
C     REPLACE SINGLE PRECISION SX BY SINGLE PRECISION SA*SX.           SS3668Ø
C     FOR I = Ø TO N-1, REPLACE SX(1+I*INCX) WITH  SA * SX(1+I*INCX)
```

```
C                                                                       SS3669Ø
      REAL SA,SX(1)                                                     SS3670Ø
      IF(N.LE.Ø)RETURN                                                  SS3671Ø
      IF(INCX.EQ.1)GOTO 2Ø                                              SS3672Ø
C                                                                       SS3673Ø
C        CODE FOR INCREMENTS NOT EQUAL TO 1.                            SS3674Ø
C                                                                       SS3675Ø
      NS = N*INCX                                                       SS3676Ø
          DO 1Ø I = 1,NS,INCX                                           SS3677Ø
          SX(I) = SA*SX(I)                                              SS3678Ø
   1Ø     CONTINUE                                                      SS3679Ø
      RETURN                                                            SS3680Ø
C                                                                       SS3681Ø
C        CODE FOR INCREMENTS EQUAL TO 1.                                SS3682Ø
C                                                                       SS3683Ø
C                                                                       SS3684Ø
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 5.   SS3685Ø
C                                                                       SS3686Ø
   2Ø M = MOD(N,5)
      IF( M .EQ. Ø ) GO TO 4Ø                                           SS3688Ø
      DO 3Ø I = 1,M                                                     SS3689Ø
        SX(I) = SA*SX(I)                                                SS3690Ø
   3Ø CONTINUE                                                          SS3691Ø
      IF( N .LT. 5 ) RETURN                                             SS3692Ø
   4Ø MP1 = M + 1                                                       SS3693Ø
      DO 5Ø I = MP1,N,5                                                 SS3694Ø
        SX(I) = SA*SX(I)                                                SS3695Ø
        SX(I + 1) = SA*SX(I + 1)                                        SS3696Ø
        SX(I + 2) = SA*SX(I + 2)                                        SS3697Ø
        SX(I + 3) = SA*SX(I + 3)                                        SS3698Ø
        SX(I + 4) = SA*SX(I + 4)                                        SS3699Ø
   5Ø CONTINUE                                                          SS3700Ø
      RETURN                                                            SS3701Ø
      END                                                               SS3702Ø


      SUBROUTINE DSCAL(N,DA,DX,INCX)                                    DS3703Ø
C                                                                       DS3704Ø
C     REPLACE DOUBLE PRECISION DX BY DOUBLE PRECISION DA*DX.            DS3705Ø
C     FOR I = Ø TO N-1, REPLACE DX(1+I*INCX) WITH  DA * DX(1+I*INCX)
C                                                                       DS3706Ø
      DOUBLE PRECISION DA,DX(1)                                         DS3707Ø
      IF(N.LE.Ø)RETURN                                                  DS3708Ø
      IF(INCX.EQ.1)GOTO 2Ø                                              DS3709Ø
C                                                                       DS3710Ø
C        CODE FOR INCREMENTS NOT EQUAL TO 1.                            DS3711Ø
C                                                                       DS3712Ø
      NS = N*INCX                                                       DS3713Ø
          DO 1Ø I = 1,NS,INCX                                           DS3714Ø
          DX(I) = DA*DX(I)                                              DS3715Ø
   1Ø     CONTINUE                                                      DS3716Ø
      RETURN                                                            DS3717Ø
C                                                                       DS3718Ø
C        CODE FOR INCREMENTS EQUAL TO 1.                                DS3719Ø
C                                                                       DS3720Ø
C                                                                       DS3721Ø
C        CLEAN-UP LOOP SO REMAINING VECTOR LENGTH IS A MULTIPLE OF 5.   DS3722Ø
C                                                                       DS3723Ø
   2Ø M = MOD(N,5)
      IF( M .EQ. Ø ) GO TO 4Ø                                           DS3725Ø
      DO 3Ø I = 1,M                                                     DS3726Ø
        DX(I) = DA*DX(I)                                                DS3727Ø
   3Ø CONTINUE                                                          DS3728Ø
      IF( N .LT. 5 ) RETURN                                             DS3729Ø
   4Ø MP1 = M + 1                                                       DS3730Ø
      DO 5Ø I = MP1,N,5                                                 DS3731Ø
        DX(I) = DA*DX(I)                                                DS3732Ø
        DX(I + 1) = DA*DX(I + 1)                                        DS3733Ø
        DX(I + 2) = DA*DX(I + 2)                                        DS3734Ø
        DX(I + 3) = DA*DX(I + 3)                                        DS3735Ø
        DX(I + 4) = DA*DX(I + 4)                                        DS3736Ø
   5Ø CONTINUE                                                          DS3737Ø
      RETURN                                                            DS3738Ø
      END                                                               DS3739Ø
```

```
      SUBROUTINE CSCAL(N,CA,CX,INCX)                                    CS374ØØ
C
C     REPLACE COMPLEX CX BY COMPLEX CA*CX.
C     FOR I = Ø TO N-1, REPLACE CX(1+I*INCX) WITH  CA * CX(1+I*INCX)
C
      COMPLEX CA,CX(1)                                                  CS3744Ø
C                                                                       CS3745Ø
      IF(N .LE. Ø) RETURN                                               CS3746Ø
      NS = N*INCX                                                       CS3747Ø
          DO 1Ø I = 1,NS,INCX                                           CS3748Ø
          CX(I) = CA*CX(I)                                              CS3749Ø
   1Ø     CONTINUE                                                      CS375ØØ
      RETURN                                                            CS3751Ø
      END                                                               CS3752Ø


      SUBROUTINE CSSCAL(N,SA,CX,INCX)                                   CS3753Ø
C
C     REPLACE COMPLEX CX BY (SINGLE PRECISION SA) * (COMPLEX CX)
C     FOR I = Ø TO N-1, REPLACE CX(1+I*INCX) WITH  SA * CX(1+I*INCX)
C
      COMPLEX CX(1)                                                     CS3757Ø
      REAL    SA                                                        CS3758Ø
C                                                                       CS3759Ø
      IF(N .LE. Ø) RETURN                                               CS3760Ø

      NS = N*INCX                                                       CS3761Ø
          DO 1Ø I = 1,NS,INCX                                           CS3762Ø
          CX(I) = SA*CX(I)                                              CS3763Ø
   1Ø     CONTINUE                                                      CS3764Ø
      RETURN                                                            CS3765Ø
      END                                                               CS3766Ø


      INTEGER FUNCTION ISAMAX(N,SX,INCX)                                IS3767Ø
C                                                                       IS3768Ø
C     FIND SMALLEST INDEX OF MAXIMUM MAGNITUDE OF SINGLE PRECISION SX.  IS3769Ø
C     ISAMAX =  FIRST I, I = 1 TO N, TO MINIMIZE  ABS(SX(1-INCX+I*INCX))
C                                                                       IS377ØØ
      REAL SX(1),SMAX,XMAG                                              IS3771Ø
      ISAMAX = Ø                                                        IS3772Ø
      IF(N.LE.Ø) RETURN                                                 IS3773Ø
      ISAMAX = 1                                                        IS3774Ø
      IF(N.LE.1)RETURN                                                  IS3775Ø
      IF(INCX.EQ.1)GOTO 2Ø                                              IS3776Ø
C                                                                       IS3777Ø
C        CODE FOR INCREMENTS NOT EQUAL TO 1.                            IS3778Ø
C                                                                       IS3779Ø
      SMAX = ABS(SX(1))                                                 IS378ØØ
      NS = N*INCX                                                       IS3781Ø
      II = 1                                                            IS3782Ø
          DO 1Ø I=1,NS,INCX                                             IS3783Ø
          XMAG = ABS(SX(I))                                             IS3784Ø
          IF(XMAG.LE.SMAX) GO TO 5                                      IS3785Ø
          ISAMAX = II                                                   IS3786Ø
          SMAX = XMAG                                                   IS3787Ø
    5     II = II + 1                                                   IS3788Ø
   1Ø     CONTINUE                                                      IS3789Ø
      RETURN                                                            IS379ØØ
C                                                                       IS3791Ø
C        CODE FOR INCREMENTS EQUAL TO 1.                                IS3792Ø
C                                                                       IS3793Ø
   2Ø SMAX = ABS(SX(1))                                                 IS3794Ø
      DO 3Ø I = 2,N                                                     IS3795Ø
         XMAG = ABS(SX(I))                                              IS3796Ø
         IF(XMAG.LE.SMAX) GO TO 3Ø                                      IS3797Ø
         ISAMAX = I                                                     IS3798Ø
         SMAX = XMAG                                                    IS3799Ø
   3Ø CONTINUE                                                          IS38ØØØ
      RETURN                                                            IS3801Ø
      END                                                               IS3802Ø
```

```
      INTEGER FUNCTION IDAMAX(N,DX,INCX)                                ID38Ø3Ø
C                                                                       ID38Ø4Ø
C     FIND SMALLEST INDEX OF MAXIMUM MAGNITUDE OF DOUBLE PRECISION DX.  ID38Ø5Ø
C     IDAMAX =  FIRST I, I = 1 TO N, TO MINIMIZE  ABS(DX(1-INCX+I*INCX))
C                                                                       ID38Ø6Ø
      DOUBLE PRECISION DX(1),DMAX,XMAG                                  ID38Ø7Ø
      IDAMAX = Ø                                                        ID38Ø8Ø
      IF(N.LE.Ø) RETURN                                                 ID38Ø9Ø
      IDAMAX = 1                                                        ID381Ø
      IF(N.LE.1)RETURN                                                  ID3811Ø
      IF(INCX.EQ.1)GOTO 2Ø                                              ID3812Ø
C                                                                       ID3813Ø
C        CODE FOR INCREMENTS NOT EQUAL TO 1.                            ID3814Ø
C                                                                       ID3815Ø
      DMAX = DABS(DX(1))                                                ID3816Ø
      NS = N*INCX                                                       ID3817Ø
      II = 1                                                            ID3818Ø
          DO 1Ø I = 1,NS,INCX                                           ID3819Ø
          XMAG = DABS(DX(I))                                            ID382ØØ
          IF(XMAG.LE.DMAX) GO TO 5                                      ID3821Ø
          IDAMAX = II                                                   ID3822Ø
          DMAX = XMAG                                                   ID3823Ø
    5     II = II + 1                                                   ID3824Ø
   1Ø     CONTINUE                                                      ID3825Ø
      RETURN                                                            ID3826Ø
C                                                                       ID3827Ø
C        CODE FOR INCREMENTS EQUAL TO 1.                                ID3828Ø
C                                                                       ID3829Ø
   2Ø DMAX = DABS(DX(1))                                                ID383ØØ
      DO 3Ø I = 2,N                                                     ID3831Ø
          XMAG = DABS(DX(I))                                            ID3832Ø
          IF(XMAG.LE.DMAX) GO TO 3Ø                                     ID3833Ø
          IDAMAX = I                                                    ID3834Ø
          DMAX = XMAG                                                   ID3835Ø
   3Ø CONTINUE                                                          ID3836Ø
      RETURN                                                            ID3837Ø
      END                                                               ID3838Ø


      INTEGER FUNCTION ICAMAX(N,CX,INCX)                                IC3839Ø
C
C      RETURNS THE INDEX OF THE COMPONENT OF CX HAVING THE              IC3841Ø
C      LARGEST SUM OF MAGNITUDES OF REAL AND IMAGINARY PARTS.           IC3842Ø
C     ICAMAX = FIRST I, I = 1 TO N, TO MINIMIZE
C        ABS(REAL(CX(1-INCX+I*INCX))) + ABS(IMAG(CX(1-INCX+I*INCX)))
C
      COMPLEX CX(1)                                                     IC3844Ø
C                                                                       IC3845Ø
      ICAMAX = Ø                                                        IC3846Ø
      IF(N.LE.Ø) RETURN                                                 IC3847Ø
      ICAMAX = 1                                                        IC3848Ø
      IF(N .LE. 1) RETURN                                               IC3849Ø
      NS = N*INCX                                                       IC385ØØ
      II = 1                                                            IC3851Ø
      SUMAX = ABS(REAL(CX(1))) + ABS(AIMAG(CX(1)))                      IC3852Ø
          DO 2Ø I=1,NS,INCX                                             IC3853Ø
          SUMRI = ABS(REAL(CX(I))) + ABS(AIMAG(CX(I)))                  IC3854Ø
          IF(SUMAX.GE.SUMRI) GO TO 1Ø                                   IC3855Ø
          SUMAX = SUMRI                                                 IC3856Ø
          ICAMAX = II                                                   IC3857Ø
   1Ø     II = II + 1                                                   IC3858Ø
   2Ø     CONTINUE                                                      IC3859Ø
      RETURN                                                            IC386ØØ
      END                                                               IC3861Ø
C                                                 BEGIN MAIN LOOP
      I = 1
   2Ø    GO TO NEXT,(3Ø, 5Ø, 7Ø, 11Ø)
   3Ø IF( DABS(DX(I)) .GT. CUTLO) GO TO 85
      ASSIGN 5Ø TO NEXT
      XMAX = ZERO
C
C                       PHASE 1.  SUM IS ZERO
C
   5Ø IF( DX(I) .EQ. ZERO) GO TO 2ØØ
      IF( DABS(DX(I)) .GT. CUTLO) GO TO 85
```

```
C
C                                  PREPARE FOR PHASE 2.
      ASSIGN 7Ø TO NEXT
      GO TO 1Ø5
C
C                                  PREPARE FOR PHASE 4.
C
  1ØØ I = J
      ASSIGN 11Ø TO NEXT
      SUM = (SUM / DX(I)) / DX(I)
  1Ø5 XMAX = DABS(DX(I))
      GO TO 115
C
C                   PHASE 2.  SUM IS SMALL.
C                             SCALE TO AVOID DESTRUCTIVE UNDERFLOW.
C
   7Ø IF( DABS(DX(I)) .GT. CUTLO ) GO TO 75
C
C                     COMMON CODE FOR PHASES 2 AND 4.
C                     IN PHASE 4 SUM IS LARGE.  SCALE TO AVOID OVERFLOW.
C
  11Ø IF( DABS(DX(I)) .LE. XMAX ) GO TO 115
         SUM = ONE + SUM * (XMAX / DX(I))**2
         XMAX = DABS(DX(I))
         GO TO 2ØØ
C
  115 SUM = SUM + (DX(I)/XMAX)**2


      GO TO 2ØØ
C
C
C                  PREPARE FOR PHASE 3.
C
   75 SUM = (SUM * XMAX) * XMAX
C     .
C
C     FOR REAL OR D.P. SET HITEST = CUTHI/N
C     FOR COMPLEX      SET HITEST = CUTHI/(2*N)
C
   85 HITEST = CUTHI/FLOAT( N )
C
C                    PHASE 3.  SUM IS MID-RANGE.  NO SCALING.
C
      DO 95 J =I,NN,INCX
      IF(DABS(DX(J)) .GE. HITEST) GO TO 1ØØ
   95    SUM = SUM + DX(J)**2
      DNRM2 = DSQRT( SUM )
      GO TO 3ØØ
C
  2ØØ CONTINUE
      I = I + INCX
      IF ( I .LE. NN ) GO TO 2Ø
C
C               END OF MAIN LOOP.
C
C               COMPUTE SQUARE ROOT AND ADJUST FOR SCALING.
C
      I = I + INCX
      IF ( I .LE. NN ) GO TO 2Ø
C
C               END OF MAIN LOOP.
C
C               COMPUTE SQUARE ROOT AND ADJUST FOR SCALING.
C
      SNRM2 = XMAX * SQRT(SUM)
   3ØØ CONTINUE
      RETURN
      END


      DOUBLE PRECISION FUNCTION DNRM2 ( N, DX, INCX)
      INTEGER          NEXT
      DOUBLE PRECISION   DX(1), CUTLO, CUTHI, HITEST, SUM, XMAX,ZERO,ONE
      DATA   ZERO, ONE /Ø.ØDØ, 1.ØDØ/
C
C     EUCLIDEAN NORM OF THE N-VECTOR STORED IN DX() WITH STORAGE
```

```
C     INCREMENT INCX .
C     IF    N .LE. Ø RETURN WITH RESULT = Ø.
C     IF N .GE. 1 THEN INCX MUST BE .GE. 1
C
C           C.L.LAWSON, 1978 JAN Ø8
C
C     FOUR PHASE METHOD     USING TWO BUILT-IN CONSTANTS THAT ARE
C     HOPEFULLY APPLICABLE TO ALL MACHINES.
C         CUTLO = MAXIMUM OF  DSQRT(U/EPS)  OVER ALL KNOWN MACHINES.
C         CUTHI = MINIMUM OF  DSQRT(V)      OVER ALL KNOWN MACHINES.
C     WHERE
C         EPS = SMALLEST NO. SUCH THAT EPS + 1. .GT. 1.
C         U   = SMALLEST POSITIVE NO.   (UNDERFLOW LIMIT)
C         V   = LARGEST  NO.            (OVERFLOW  LIMIT)
C
C     BRIEF OUTLINE OF ALGORITHM..
C
C     PHASE 1    SCANS ZERO COMPONENTS.
C     MOVE TO PHASE 2 WHEN A COMPONENT IS NONZERO AND .LE. CUTLO
C     MOVE TO PHASE 3 WHEN A COMPONENT IS .GT. CUTLO
C     MOVE TO PHASE 4 WHEN A COMPONENT IS .GE. CUTHI/M
C     WHERE M = N FOR X() REAL AND M = 2*N FOR COMPLEX.
C
C     VALUES FOR CUTLO AND CUTHI..
C     FROM THE ENVIRONMENTAL PARAMETERS LISTED IN THE IMSL CONVERTER
C     DOCUMENT THE LIMITING VALUES ARE AS FOLLOWS..
C     CUTLO, S.P.   U/EPS = 2**(-1Ø2) FOR  HONEYWELL.  CLOSE SECONDS ARE
C                   UNIVAC AND DEC AT 2**(-1Ø3)
C                   THUS CUTLO = 2**(-51) = 4.44Ø89E-16
C     CUTHI, S.P.   V = 2**127 FOR UNIVAC, HONEYWELL, AND DEC.
C                   THUS CUTHI = 2**(63.5) = 1.3Ø438E19
C     CUTLO, D.P.   U/EPS = 2**(-67) FOR HONEYWELL AND DEC.
C                   THUS CUTLO = 2**(-33.5) = 8.23181D-11

C     CUTHI, D.P.   SAME AS S.P.  CUTHI = 1.3Ø438D19
C     DATA CUTLO, CUTHI / 8.232D-11,  1.3Ø4D19 /
C     DATA CUTLO, CUTHI / 4.441E-16,  1.3Ø4E19 /
      DATA CUTLO, CUTHI / 8.232D-11,  1.3Ø4D19 /
C
      IF(N .GT. Ø) GO TO 1Ø
         DNRM2  = ZERO
         GO TO 3ØØ
C
   1Ø ASSIGN 3Ø TO NEXT
      SUM = ZERO
      NN = N * INCX



      DOUBLE PRECISION FUNCTION DQDOTI(N,DB,QC,DX,INCX,DY,INCY)
C     D.P. DOT PRODUCT WITH EXTENDED PRECISION ACCUMULATION (AND RESULT)
C     QC AND DQDOTI ARE SET = DB + SUM FOR I = Ø TO N-1 OF
C       DX(LX+I*INCX) * DY(LY+I*INCY),  WHERE QC IS AN EXTENDED
C       PRECISION RESULT WHICH CAN BE USED AS INPUT TO DQDOTA,
C       AND LX = 1 IF INCX .GE. Ø, ELSE LX = (-INCX)*N, AND LY IS
C       DEFINED IN A SIMILAR WAY USING INCY. THE MP PACKAGE BY
C       RICHARD P. BRENT IS USED FOR THE EXTENDED PRECISION ARITHMETIC.
C
C     FRED T. KROGH,  JPL,  1977,  JUNE 1
C2
      DOUBLE PRECISION DX(1), DY(1), DB
      INTEGER  QC(1Ø), QX(1Ø), QY(1Ø)
C     THE COMMON BLOCK FOR THE MP PACKAGE (MODIFIED TO GIVE IT A NAME)
      COMMON /MPCOM/  MPB, MPT, MPM, MPLUN, MPMXR, MPR(12)
      DATA  I1 / Ø /
C     IF I1 IS Ø THE MP PACKAGE MUST BE INITIALIZED (MPBLAS SETS I1 = 1)
      IF (I1 .EQ. Ø) CALL MPBLAS(I1)
      QC(1) = Ø
      IF (DB .EQ. Ø.DØ) GO TO 6Ø
      CALL MPCDM(DB, QX)
      CALL MPADD(QC, QX, QC)
   6Ø IF (N .EQ. Ø) GO TO 8Ø
      IX = 1
      IY = 1
      IF (INCX .LT. Ø) IX = (-N + 1) * INCX + 1
      IF (INCY .LT. Ø) IY = (-N + 1) * INCY + 1
      DO  7Ø  I = 1,N
```

```
            CALL MPCDM(DX(IX), QX)
            CALL MPCDM(DY(IY), QY)
            CALL MPMUL(QX, QY, QX)
            CALL MPADD(QC, QX, QC)
            IX = IX + INCX
            IY = IY + INCY
         7Ø CONTINUE
         8Ø CALL MPCMD(QC, DQDOTI)
            RETURN
            END


            SUBROUTINE DROTM (N,DX,INCX,DY,INCY,DPARAM)                          DR2947Ø
      C                                                                          DR2948Ø
      C     APPLY THE MODIFIED GIVENS TRANSFORMATION, H, TO THE 2 BY N MATRIX    DR2949Ø
      C                                                                          DR2950Ø
      C     (DX**T) , WHERE **T INDICATES TRANSPOSE. THE ELEMENTS OF DX ARE IN
      C     (DY**T)
      C
      C     DX(LX+I*INCX), I = Ø TO N-1, WHERE LX = 1 IF INCX .GE. Ø, ELSE
      C     LX = (-INCX)*N, AND SIMILARLY FOR SY USING LY AND INCY.
      C     WITH DPARAM(1)=DFLAG, H HAS ONE OF THE FOLLOWING FORMS..             DR2955Ø
      C                                                                          DR2956Ø
      C     DFLAG=-1.DØ     DFLAG=Ø.DØ        DFLAG=1.DØ     DFLAG=-2.DØ       DR2957Ø
      C                                                                          DR2958Ø
      C       (DH11  DH12)    (1.DØ  DH12)    (DH11  1.DØ)    (1.DØ  Ø.DØ)     DR2959Ø
      C     H=(           )   (           )   (           )   (           )      DR2960Ø
      C       (DH21  DH22),   (DH21  1.DØ),   (-1.DØ DH22),   (Ø.DØ  1.DØ).    DR2961Ø
      C     SEE DROTMG FOR A DESCRIPTION OF DATA STORAGE IN DPARAM.
      C                                                                          DR2962Ø
            DOUBLE PRECISION DFLAG,DH12,DH22,DX,TWO,Z,DH11,DH21,                 DR2963Ø
           1 DPARAM,DY,W,ZERO                                                    DR2964Ø
            DIMENSION DX(1),DY(1),DPARAM(5)                                      DR2965Ø
            DATA ZERO,TWO/Ø.DØ,2.DØ/                                             DR2966Ø
      C                                                                          DR2967Ø
            DFLAG=DPARAM(1)                                                      DR2968Ø
            IF(N .LE. Ø .OR.(DFLAG+TWO.EQ.ZERO)) GO TO 14Ø                       DR2969Ø
                IF(.NOT.(INCX.EQ.INCY.AND. INCX .GT.Ø)) GO TO 7Ø                 DR2970Ø
      C                                                                          DR2971Ø
                     NSTEPS=N*INCX                                               DR2972Ø
                     IF(DFLAG) 5Ø,1Ø,3Ø                                          DR2973Ø
         1Ø          CONTINUE                                                    DR2974Ø
                     DH12=DPARAM(4)                                              DR2975Ø
                     DH21=DPARAM(3)                                              DR2976Ø
                          DO 2Ø I=1,NSTEPS,INCX                                  DR2977Ø
                          W=DX(I)                                                DR2978Ø
                          Z=DY(I)                                                DR2979Ø
                          DX(I)=W+Z*DH12                                         DR2980Ø
                          DY(I)=W*DH21+Z                                         DR2981Ø
         2Ø               CONTINUE                                               DR2982Ø
                     GO TO 14Ø                                                   DR2983Ø
         3Ø          CONTINUE                                                    DR2984Ø
                     DH11=DPARAM(2)                                              DR2985Ø
                     DH22=DPARAM(5)                                              DR2986Ø
                          DO 4Ø I=1,NSTEPS,INCX                                  DR2987Ø
                          W=DX(I)                                                DR2988Ø
                          Z=DY(I)                                                DR2989Ø
                          DX(I)=W*DH11+Z                                         DR2990Ø
                          DY(I)=-W+DH22*Z                                        DR2991Ø
         4Ø               CONTINUE                                               DR2992Ø
                     GO TO 14Ø                                                   DR2993Ø
         5Ø          CONTINUE                                                    DR2994Ø
                     DH11=DPARAM(2)                                              DR2995Ø
                     DH12=DPARAM(4)                                              DR2996Ø
                     DH21=DPARAM(3)                                              DR2997Ø
                     DH22=DPARAM(5)                                              DR2998Ø
                          DO 6Ø I=1,NSTEPS,INCX                                  DR2999Ø
                          W=DX(I)                                                DR3ØØØØ
                          Z=DY(I)                                                DR3Ø01Ø
                          DX(I)=W*DH11+Z*DH12                                    DR3Ø02Ø
                          DY(I)=W*DH21+Z*DH22                                    DR3Ø03Ø
         6Ø               CONTINUE                                               DR3Ø04Ø
                     GO TO 14Ø                                                   DR3Ø05Ø
         7Ø      CONTINUE                                                       DR3Ø06Ø
                 KX=1                                                           DR3Ø07Ø
```

```
      KY=1                                                          DR3ØØ8Ø
      IF(INCX .LT. Ø) KX=1+(1-N)*INCX                               DR3ØØ9Ø
      IF(INCY .LT. Ø) KY=1+(1-N)*INCY                               DR3Ø1ØØ
C                                                                   DR3Ø11Ø
      IF(DFLAG)12Ø,8Ø,1ØØ                                           DR3Ø12Ø
 8Ø   CONTINUE                                                      DR3Ø13Ø
      DH12=DPARAM(4)                                                DR3Ø14Ø
      DH21=DPARAM(3)                                                DR3Ø15Ø
           DO 9Ø I=1,N                                              DR3Ø16Ø
           W=DX(KX)                                                 DR3Ø17Ø
           Z=DY(KY)                                                 DR3Ø18Ø
           DX(KX)=W+Z*DH12                                          DR3Ø19Ø
           DY(KY)=W*DH21+Z                                          DR3Ø2ØØ
           KX=KX+INCX                                               DR3Ø21Ø
           KY=KY+INCY                                               DR3Ø22Ø
 9Ø        CONTINUE                                                 DR3Ø23Ø
      GO TO 14Ø                                                     DR3Ø24Ø
1ØØ   CONTINUE                                                      DR3Ø25Ø
      DH11=DPARAM(2)                                                DR3Ø26Ø
      DH22=DPARAM(5)                                                DR3Ø27Ø
           DO 11Ø I=1,N                                             DR3Ø28Ø
           W=DX(KX)                                                 DR3Ø29Ø
           Z=DY(KY)                                                 DR3Ø3ØØ
           DX(KX)=W*DH11+Z                                          DR3Ø31Ø
           DY(KY)=-W+DH22*Z                                         DR3Ø32Ø
           KX=KX+INCX                                               DR3Ø33Ø
           KY=KY+INCY                                               DR3Ø34Ø
11Ø        CONTINUE                                                 DR3Ø35Ø
      GO TO 14Ø                                                     DR3Ø36Ø
12Ø   CONTINUE                                                      DR3Ø37Ø
      DH11=DPARAM(2)                                                DR3Ø38Ø
      DH12=DPARAM(4)                                                DR3Ø39Ø
      DH21=DPARAM(3)                                                DR3Ø4ØØ
      DH22=DPARAM(5)                                                DR3Ø41Ø
           DO 13Ø I=1,N                                             DR3Ø42Ø
           W=DX(KX)                                                 DR3Ø43Ø
           Z=DY(KY)                                                 DR3Ø44Ø
           DX(KX)=W*DH11+Z*DH12                                     DR3Ø45Ø
           DY(KY)=W*DH21+Z*DH22                                     DR3Ø46Ø
           KX=KX+INCX                                               DR3Ø47Ø
           KY=KY+INCY                                               DR3Ø48Ø
13Ø        CONTINUE                                                 DR3Ø49Ø
14Ø   CONTINUE                                                      DR3Ø5ØØ
      RETURN                                                        DR3Ø51Ø
      END                                                           DR3Ø52Ø
```

ACM Transactions on Mathematical Software, Vol. 8, No. 4, December 1982, Pages 403–404.

REMARK ON ALGORITHM 539

Basic Linear Algebra Subprograms for Fortran Usage [C.L. Lawson, R.J. Hanson, D.R. Kincaid, and F.T. Krogh, *ACM Trans. Math. Softw. 5*, 3 (Sept. 1979), 324–325]

David S. Dodson and Roger G. Grimes [Received 11 May 1982; revised 25 August 1982; accepted 17 September 1982]

Boeing Computer Services Company, Mail Stop 9C-01, 565 Andover Park West, Tukwila, WA 98188.

The companion [5] to Algorithm 539 (the BLAS) contains two errors which we discovered in the preparation of [1].

The more serious error occurs in the mathematical specifications for subroutines **SROTG** and **DROTG**, which construct Givens plane rotations. For convenience of reference, we include the relevant specifications from [5].

Given a and b, each of these subroutines computes

$$\sigma = \begin{cases} \operatorname{sgn}(a) & \text{if } |a| > |b|, \\ \operatorname{sgn}(b) & \text{if } |b| \geq |a|, \end{cases} \qquad r = \sigma(a^2 + b^2)^{1/2}, \qquad \text{(1a,b)}$$

Table I. Status of BLAS Implementations

Source	SROTG and DROTG		
	Reference	Documentation	Code
Boeing Computer Services	[1]	Correct	Correct
Dongarra et al.	[2]	Incorrect	Incorrect
Floating Point Systems	[3]	Incorrect	Incorrect
IMSL	[4]	Incorrect	Correct
Lawson et al.	[5]	Incorrect	Correct
Petersen	[6]	Correct	Incorrect

$$= \begin{cases} a/r & \text{if } r \neq 0, \\ 1 & \text{if } r = 0, \end{cases} \qquad s = \begin{cases} b/r & \text{if } r \neq 0, \\ 1 & \text{if } r = 0, \end{cases} \tag{1c,d}$$

$$z = \begin{cases} s & \text{if } |a| > |b|, \\ 1/c & \text{if } |b| \geq |a| \quad \text{and} \quad c \neq 0, \\ 1 & \text{if } c = 0. \end{cases} \tag{1e}$$

If the user later wishes to reconstruct c and s from z, it can be done as follows:

If $z = 1$ set $c = 0$ and $s = 1$.

If $|z| < 1$ set $c = (1 - z^2)^{1/2}$ and $s = z$. (2)

If $|z| > 1$ set $c = 1/z$ and $s = (1 - c^2)^{1/2}$.

The problem occurs in the computation of z. In particular, when $a = b = 0$, (1a–e) yield the results $r = 0$, $c = 1$, $s = 0$, and $z = 1$. However, when c and s are reconstructed from z using (2), the incorrect values $c = 0$ and $s = 1$ result. This discrepancy can be resolved only by changing the computation of z, as $z = 1$ can result both when $c = 0$ (when $a = 0$ and $b \neq 0$) and when $c = 1$ (when $a = b = 0$).

Stewart [7] intended that z represent the smaller in magnitude of c and s and that it indicate which one of c and s is the smaller, since the magnitude of the larger can be reconstructed stably from the smaller. He expressly omitted consideration of the point $a = b = 0$, which is the only point at which (1e) fails. We propose that z be computed directly from c and s. Two observations simplify matters. First, the larger in magnitude of c and s is positive: $|s| < |c|$ implies $c > 0$ and $0 < |c| \leq |s|$ implies $s > 0$. Second, if $c = 0$, then $s = 1$. Thus the value z can be redefined as follows:

$$z = \begin{cases} s & \text{if } |s| < c \quad \text{or} \quad c = 0, \\ 1/c & \text{if } 0 < |c| \leq s. \end{cases} \tag{3}$$

This definition of z in (3) differs from (1a–e) only at the single point $a = b = 0$, where $z = 0$ results. For $z = 0$, the reconstruction formulas (2) give the correct values, $c = 1$ and $s = 0$.

Even though the original mathematical specifications for subroutines **SROTG** and **DROTG** are incorrect, the FORTRAN and assembly language versions of **SROTG** and **DROTG**, available from the ACM Algorithm Distribution Service as part of Algorithm 539, do conform to the correct mathematical specification, as given in this Remark.

BLAS documentation has appeared in several publications, and BLAS code is available for use or distribution from several sources. Table I lists the status of some of these BLAS implementations.

A second error, more innocuous than the above, involves the dimensions of the arrays passed to the BLAS. In several places, the length is incorrectly specified as max(**1, N*|INCX|**). The correct length is max(**1, 1+(N − 1)*|INCX|**). This error occurs near the beginnings of Sections 5 and 7, and would be unobtrusive if Section 7 did not contain the word "precisely."

ACKNOWLEDGMENT

We thank Dr. C. L. Lawson for his helpful comments, which have led to a redefinition of z that is superior to what we had originally suggested.

REFERENCES

1. Boeing Computer Services Company *MAINSTREAM-EKS/VSP CRAYPACK Supplement to BCSLIB Users Manual.* Pub. no. 10208-2024, Seattle, Wash., 1982.
2. Dongarra, J.J., Bunch, J.R., Moler, C.B., and Stewart, G.W. *LINPACK Users' Guide.* Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1979.
3. Floating Point Systems, Inc. *APMATH64 Manual.* Pub. no. 860-7482-000B, Portland, Ore., 1982.
4. International Mathematical and Statistical Libraries, Inc. *IMSL Library Reference Manual,* 8th ed., Houston, Tex., 1980.
5. Lawson, C.L., Hanson, R.J., Kincaid, D.R., and Krogh, F.T. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Softw. 5,* 3 (Sept. 1979), 308–323.
6. Petersen, W.P. CRAY-1 basic linear algebra subprograms for CFT usage. Pub. no. 2240208, Cray Research, Inc., Minneapolis, Minn., 1979.
7. Stewart, G.W. The economical storage of plane rotations. *Numer. Math. 25,* 2 (1976), 137–139.

ALGORITHM 540
PDECOL, General Collocation Software for Partial Differential Equations [D3]

N. K. MADSEN
Lawrence Livermore Laboratory
and
R. F. SINCOVEC
Kansas State University

Key Words and Phrases: collocation methods, partial differential equations, numerical software, method of lines
CR Categories: 3.20, 3.22, 4.0, 5.17
Language: Fortran

DESCRIPTION

1. Introduction

The basic purpose of this paper is to describe and discuss PDECOL, which is a new computer software package for numerically solving coupled systems of nonlinear partial differential equations (PDE's) in one space and one time dimension. The package implements finite element collocation methods based on piecewise polynomials for the spatial discretization techniques. The time integration process is then accomplished by widely acceptable procedures [7] which are generalizations of the usual methods for treating time dependent partial differential equations.

PDECOL is unique because of its flexibility both in the class of problems it addresses and in the variety of methods it provides for use in the solution process. High order methods (as well as low order ones) are readily available for use in both the spatial and time discretization procedures. The time integration methods used feature automatic time step size and integration formula order selection so as to efficiently solve the problem at hand and yet achieve a user specified time integration error level.

PDECOL consists of a collection of 19 subroutines written in reasonably standard Fortran, and therefore is quite portable and can be readily used on almost any capable computer with a Fortran compiler. No special hardware features are required.

Received 25 June 1976 and 1 August 1977.

This work was performed under the auspices of the U.S. Energy Research and Development Administration under Contract W-7405-Eng-48.
Authors' present addresses: N.K. Madsen, Lawrence Livermore Laboratory, Livermore, CA 94550; R.F. Sincovec, Boeing Computer Services, Co., Energy Technology Applications, Mail Stop 9C-01, P.O. Box 24346, Seattle, WA 98124.

PDECOL is designed to solve broad classes of difficult systems of partial differential equations that describe physical processes. This package should be of interest to almost anyone involved in scientific and engineering simulations, calculations, and model development. It should also be of interest to those involved in mathematical software development and, in particular, to those concerned with software development for partial differential equations.

One of our main objectives in producing PDECOL was to make available a package which will allow users to try out a variety of collocation techniques on a variety of problems with little program development required. As is mentioned in the text, there are over 3000 different potentially useful PDE methods in PDECOL. Basically, we are relying on others to produce the final judgments as to the overall effectiveness of collocation for their problems. We feel from our experience that collocation looks attractive enough to warrant serious consideration by others—hence, the package PDECOL.

2. Class of Problems

PDECOL is designed to solve the general system of *NPDE* nonlinear partial differential equations of at most second order on the interval $[x_L, x_R]$ for $t \geq t_0$ which is of the form

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{f}(t, x, \mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}), \tag{2.1}$$

where

$$\begin{aligned} \mathbf{u} &= (u_1, u_2, \ldots, u_{NPDE}), \\ \mathbf{u}_x &= \left(\frac{\partial u_1}{\partial x}, \frac{\partial u_2}{\partial x}, \ldots, \frac{\partial u_{NPDE}}{\partial x}\right), \\ \mathbf{u}_{xx} &= \left(\frac{\partial^2 u_1}{\partial x^2}, \frac{\partial^2 u_2}{\partial x^2}, \ldots, \frac{\partial^2 u_{NPDE}}{\partial x^2}\right). \end{aligned} \tag{2.2}$$

Each u_k is a function of the scalar quantities t and x, $k = 1, 2, \ldots, NPDE$. In (2.1) $\mathbf{f}$ represents an arbitrary vector valued function whose *NPDE* components define the respective partial differential equations of the PDE system. As there is no explicit requirement that $\mathbf{f}$ actually depends on $\mathbf{u}_x$ and/or $\mathbf{u}_{xx}$, any particular equation in (2.1) may actually be an ordinary differential equation (ODE), a first-order PDE, or a second-order PDE.

Depending on the particular type of equation, 0, 1, or 2 boundary conditions may be required for each equation in the system (2.1). These are imposed at x_L and/or x_R (or not imposed at all in the case of no condition) and must be of the form

$$\mathbf{b}(\mathbf{u}, \mathbf{u}_x) = \mathbf{z}(t), \tag{2.3}$$

where $\mathbf{b}$ and $\mathbf{z}$ are arbitrary vector valued functions with *NPDE* components and $\mathbf{u}$, $\mathbf{u}_x$, and t are as above. We make the basic assumption that these boundary conditions (2.3) must be consistent with the initial conditions (2.4) which are described next.

Each solution component u_k is assumed to be a known function of x at the initial time $t = t_0$. That is,

$$u_k(t_0, x) = \phi_k(x), \qquad k = 1, 2, \ldots, NPDE, \tag{2.4}$$

where each $\phi_k,(x)$ is a known function of x. The initial condition functions must be consistent with the boundary conditions (2.3), (i.e. the initial condition functions must satisfy the boundary conditions for $t = t_0$).

We assume that all functions are continuous in t and at least piecewise continuous in x. With some understanding of the internal workings of the package, it is possible to solve problems with inconsistent boundary and initial conditions

or problems with jump discontinuities in the boundary conditions with respect to t.

It is obvious that the possible PDE problems described above are sufficiently general to include many systems for which solutions may not exist or may not be unique. *It must be, of course, the user's responsibility to define a mathematically meaningful PDE problem.*

3. Defining the PDE Problem

The PDE system described above is completely specified if one defines: *NPDE*; the interval $[x_L, x_R]$; the initial time t_0; the vector functions **f**, **b**, and **z**; and the initial condition functions $\phi_k(x)$, $k = 1, 2, \ldots,$ *NPDE*. The PDE problem can be properly defined for solution by PDECOL by constructing a main program and three subprograms, F, BNDRY, and UINIT, which specify the above-mentioned quantities.

Main Program. The construction of the main program that a user is required to supply is usually straightforward. The main program serves four basic purposes. First, initialization of the calling arguments for PDECOL must be performed. Second, the main program must call the package. Third, it should provide logic to detect possible error returns from PDECOL. Fourth, the main program should perform the desired output of the computed results. A typical main program is shown in a later section for one numerical example.

Required User Supplied Subroutines. The user is required to construct three subroutines which define the form of the PDE problem. Stated briefly, the purposes of these routines, F, BNDRY, and UINIT, are as follows. For given input values of t and x and corresponding input values of $\mathbf{u}$, $\mathbf{u}_x$, and $\mathbf{u}_{xx}$ which are associated with this time and spatial position: subroutine F is to compute appropriate values for the functions f_k in (2.1); subroutine BNDRY is to compute appropriate values for the derivatives of the boundary condition functions b_k and z_k in (2.3) at the left or right boundary as determined by the value of x (see (4.3) and (4.4)); and subroutine UINIT is to compute initial condition function values (2.4).

The subroutines F, BNDRY, and UINIT are usually easily constructed and an example showing these routines is contained in a later section. More specific comments are given in the Algorithm at the end of this paper.

4. Methods Used

The software package PDECOL is based on the method of lines [7, 8, 11] and uses a finite element collocation procedure (with piecewise polynomials as the trial space) for the discretization of the spatial variable x. The collocation procedure reduces the PDE system to a semidiscrete system (actually an initial-value ODE system), which then depends only on the time variable t. The time integration is then accomplished by use of slightly modified standard techniques [5, 6, 7], which will be discussed briefly in a later section.

Piecewise Polynomials. The user has the opportunity to specify the piecewise polynomial space which is to be used to compute his approximate solution. In selecting this space the order, *KORD*, of the polynomials to be used must first be specified (*KORD* = polynomial degree + 1). Next, the number of pieces (intervals), *NINT*, into which the spatial domain $[x_L, x_R]$ is to be divided is chosen. The *NINT* + 1 distinct breakpoints of the domain must be defined and set into the array *XBKPT* in strictly increasing order, i.e.

$$x_L = XBKPT(1) < XBKPT(2) < \cdots < XBKPT(NINT + 1) = x_R.$$

The approximate solution at any time, t, will be a polynomial of order *KORD* in each subinterval $[XBKPT(i), XBKPT(i + 1)]$, $i = 1, 2, \ldots,$ *NINT*. The number of continuity conditions, *NCC*, to be imposed on the polynomial pieces across all of the interior breakpoints is the last piece of user supplied data which is required

to uniquely determine the desired piecewise polynomial space. For example, *NCC* = 2 would require that the approximate solution (made up of the separate polynomial pieces) and its first spatial derivative be continuous at the breakpoints and hence on the entire domain $[x_L, x_R]$. *NCC* = 3 would require in addition that the second spatial derivative be continuous. The dimension of this linear space is known and finite and is $NCPTS = KORD*NINT - NCC*(NINT - 1)$. The well-known B-spline basis [2] for this space is used by PDECOL and it consists of *NCPTS* known piecewise polynomial functions $\Phi_i(x)$, $i = 1, 2, \ldots, NCPTS$, which do not depend on the time variable t. Use of the B-spline basis results in banded (in contrast to full) matrix problems to be solved in the package. The computer program requires $3 \le KORD \le 20$, $1 < NCC < KORD$, and $NINT \ge 1$.

Collocation over Piecewise Polynomials. The basic assumption made is that at any given time t, each approximate solution component, u_k, is a piecewise polynomial in the user specified space and hence can be written in terms of the B-spline basis functions as

$$u_k(t, x) = \sum_{i=1}^{NCPTS} c_{i,k}(t)\, \Phi_i(x), \qquad k = 1, 2, \ldots, NPDE. \tag{4.1}$$

The unknown coefficients $c_{i,k}$ depend only on the time t, and the *known* basis functions Φ_i depend on x. The semidiscrete equations (actually ordinary differential equations) which determine these coefficients, $c_{i,k}$, for $i = 1, 2, \ldots, NCPTS$ and $k = 1, 2, \ldots, NPDE$, are obtained by collocating, i.e. by requiring the approximate $u_k(t, x)$ in (4.1) to satisfy the PDE's (2.1) and the boundary conditions (2.3) exactly, at a set of *NCPTS* collocation points.

To be more specific, we choose *NCPTS* collocation points such that

$$x_L = \xi_1 < \xi_2 < \cdots < \xi_{NCPTS} = x_R$$

and $\Phi_i(\xi_i) \neq 0$ for $i = 1, 2, \ldots, NCPTS$. Then, substituting (4.1) into (2.1) and requiring (2.1) to be valid at the interior collocation points gives

$$\sum_{i=1}^{NCPTS} \Phi_i(\xi_j) \frac{dc_{i,k}}{dt} = f_k(t, \xi_j, \mathbf{u}(t, \xi_j), \mathbf{u}_x(t, \xi_j), \mathbf{u}_{xx}(t, \xi_j)),$$
$$j = 2, 3, \ldots, NCPTS - 1, \quad k = 1, 2, \ldots, NPDE. \tag{4.2}$$

To determine the equations corresponding to $j = 1$ and $j = NCPTS$, we form equations which depend on the type of boundary condition. It will suffice to show the technique we use for the left boundary, $x = x_L$ where $j = 1$, since those for the right boundary, $x = x_R$ where $j = NCPTS$, are completely analogous.

Normally, we form an ODE corresponding to the point $x = x_L$ by differentiating the boundary conditions (2.3) with respect to t, which gives

$$\sum_{j=1}^{NPDE} \left\{ \frac{\partial b_k}{\partial u_j} \frac{\partial u_j}{\partial t} + \frac{\partial b_k}{\partial u_{x_j}} \frac{\partial u_{x_j}}{\partial t} \right\} = \frac{dz_k}{dt}. \tag{4.3}$$

Substituting (4.1) into (4.3) and using the facts [1, 3] that $\Phi_1(x_L) \neq 0$, $\Phi_i(x_L) = 0$, $i = 2, 3, \ldots, NCPTS$, and that $\Phi_1'(x_L) \neq 0$, $\Phi_2'(x_L) \neq 0$, $\Phi_i'(x_L) = 0$, $i = 3, 4, \ldots, NCPTS$ gives the appropriate ODE:

$$\sum_{j=1}^{NPDE} \left\{ \frac{\partial b_k}{\partial u_j} \Phi_1(x_L) + \frac{\partial b_k}{\partial u_{x_j}} \Phi_1'(x_L) \right\} \frac{dc_{1,j}}{dt} + \sum_{j=1}^{NPDE} \left\{ \frac{\partial b_k}{\partial u_{x_j}} \Phi_2'(x_L) \right\} \frac{dc_{2,j}}{dt} = \frac{dz_k}{dt}. \tag{4.4}$$

In the special case when no boundary condition is desired for the $k = k_0$ equation, we simply collocate in the usual manner at the boundary point and obtain (4.2) with $j = 1$ and $k = k_0$ for our equation corresponding to the point $x = x_L$.

Combining the boundary condition equations with (4.2) yields a semidiscrete system of $N = NPDE*NCPTS$ time dependent ordinary differential equations which have the form

$$A\frac{d\mathbf{c}}{dt} = \mathbf{g}(t, \mathbf{c}). \tag{4.5}$$

The matrix A in (4.5) and the Jacobian matrix of $\mathbf{g}$, $\partial\mathbf{g}/\partial\mathbf{c}$, are matrices with a maximum bandwidth of $2*(KORD-1)*NPDE-1$ except when $KORD=3$ and an equation with no boundary condition at one endpoint exists. Except for the first and last block rows, all of the entries of A are simply basis function values at the collocation points. The first and last block rows of A consist of the appropriate boundary condition equation coefficients from (4.4) or (4.2). Since at any point at most *KORD* basis functions have nonzero values, each block row of A consists of *KORD NPDE* by *NPDE* matrices.

We remind the reader that the unknowns in (4.5) which are actually computed by PDECOL are the *basis function coefficients* in (4.1) and *not* the actual approximate solution values. However, with a knowledge of these coefficients, we can then easily evaluate the approximate solution values by using (4.1).

Accuracy Considerations. There are two sources of error in the approximate solutions generated by PDECOL. The first is due to the time discretization methods used and the second is due to the collocation spatial discretization technique. PDECOL attempts to control the time discretization error and to maintain it below a user specified level by dynamically selecting appropriate time step sizes and time integration formulas. We refer the reader to the package documentation for the details on how this is accomplished. Control of the error introduced by the spatial discretization is a more difficult problem, and we make the following observations.

Piecewise polynomials of order *KORD* have the approximation property [13] that sufficiently smooth functions can be approximated by these polynomials such that the spatial errors in the approximation are proportional to h^{KORD}, where $h=\max_i[XBKPT(i+1)-XBKPT(i)]$. Derivatives of order j of these smooth functions are also approximated with spatial errors which are proportional to h^{KORD-j}. We can use these approximation properties to estimate the expected orders of accuracy (for the *spatial* discretization) of the collocation techniques implemented in PDECOL. In particular, when using a piecewise polynomial space of order *KORD*, since eq. (2.1) involves second-order spatial derivatives, we expect PDECOL to generate approximate solutions with *spatial* errors which are proportional to h^{KORD-2}. However, for the special spaces with $KORD>3$ and $NCC=2$, a special choice of collocation points (Gauss-Legendre quadrature points in each subinterval) generates approximate solutions with *spatial* errors which are proportional to h^{KORD} for certain classes of PDE problems [4].

We emphasize that the piecewise polynomial space used in PDECOL (*which is selected by the user*) will determine the magnitude of the spatial discretization errors in the computed approximate solution. *The package has no control over errors introduced by the user's choice of the piecewise polynomial space.*

5. Limitations and Use of PDECOL

Of course, it goes without saying that no one program such as PDECOL will solve all PDE problems. There are several reasons for this, and we will enumerate them.

First, in order to be "solvable" a PDE problem should be properly posed in a mathematical sense. There is nothing that PDECOL can do to insure or detect that a user's problem is properly posed and so we reiterate that this responsibility must fall back to the user. Discussion of what constitutes a properly posed problem is beyond the scope and intent of this paper and we refer the interested reader to any good PDE textbook.

Second, PDECOL (and almost any general purpose software package) imposes certain restrictions on the classes of problems it allows. These restrictions are enumerated in Section 2. Perhaps the most unpleasant of these restrictions is the form of the boundary conditions (2.3) and the fact that the boundary conditions must be consistent with the initial conditions. Maintaining high order approxi-

mations at the boundary is a difficult problem in solving PDE's, and the above restrictions result from our approach to accomplish this.

All problems do not fit these restrictions. For example, certain hyperbolic systems of PDE's require (for stability) that the boundary conditions be imposed through the use of characteristic transformations at the boundary. We know of no way to implement the use of characteristic variables for the boundary conditions with the current structure of PDECOL. Another limitation is that PDECOL is restricted to problems in at most one space dimension. The extension to higher dimensions of the method of lines approach with either collocation or finite differences is conceptually quite simple. The basic difficulty arises from the fact that this basic approach yields a *fully implicit* discretization method. This, of course, implies that a possibly very large matrix problem must be solved. We feel certain that there are classes of PDE problems where fully implicit methods will be necessary, and for these problems extensions of these current techniques would prove fruitful. For other classes of problems, splitting or alternating direction implicit techniques may be much more efficient.

If a user's problem does not fit the prescription of the package, he has the following options: (a) find another package, (b) modify the existing package to suit his needs, or (c) modify or transform the problem to fit the package. The complexity of PDECOL and the limit of space precludes presenting details on how to modify PDECOL to accommodate problems which may not satisfy the above restrictions.

Third, the methods implemented in a program may not be adequate for a particular problem. Collocation methods (as implemented in PDECOL) will not work for all problems. For example, some PDE problems are quite sensitive to a particular conservation law being satisfied by the discrete approximation (conservation of mass, energy, particles, etc.). Collocation techniques are inherently nonconservative and difficulties can be expected if such sensitivities exist for a problem.

Summarizing, our experience in using general purpose software such as PDECOL over several years has led us to believe that general purpose software is somewhat more difficult to use because of its generality. Therefore, the probability for errors and misuse is greater and so its primary advantage (its general nature) also becomes a disadvantage. We have found that users have difficulties primarily for three reasons: (1) their problem is not well posed, (2) the methods in the general package will not work for their problem, and (3) the user has made programming errors or misused the package. Perhaps 99 percent of all difficulties fall into the third category.

6. Structure of PDECOL

Since the methods in PDECOL are based on the method of lines, PDECOL is structured much like some of the recently developed integrators for ordinary differential equations. This structure includes a driver subroutine, a core integrator which advances the time by a single time step, and miscellaneous routines which assist in setting up and solving nonlinear and linear equations.

PDECOL is somewhat more complicated than the typical ODE integrator since: (a) PDE's are being solved, and (b) a finite element method requiring the evaluation of the piecewise polynomials (4.1) is being used. The interconnections and lines of communication in PDECOL are quite complicated and difficult to understand when viewed as a whole. However, when the driver program and the core integrator are isolated, the picture becomes more clear. Figures 1 and 2 illustrate all of the package and user routines and their interconnections. Some of the routines are duplicated in each figure. A brief summary of the basic functions of all of the package subroutines is contained in the Algorithm. We will now describe the principal routines in PDECOL.

Main Components. PDECOL. This routine is a driver for the entire package. It serves the following purposes. The locations and lengths of the storage arrays

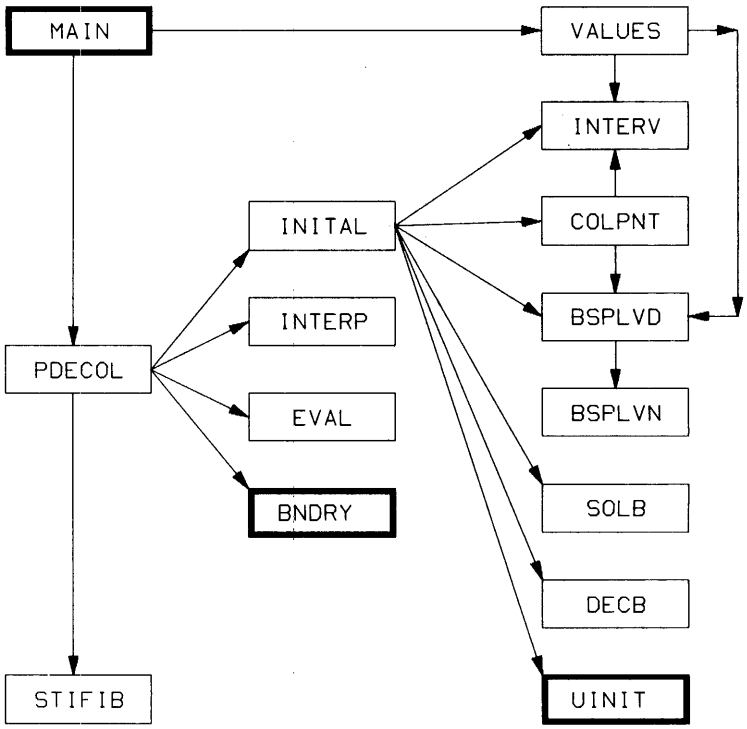


Fig. 1. Lines of communication between user routines and PDECOL package routines: Part 1, user routines are indicated by heavy borders

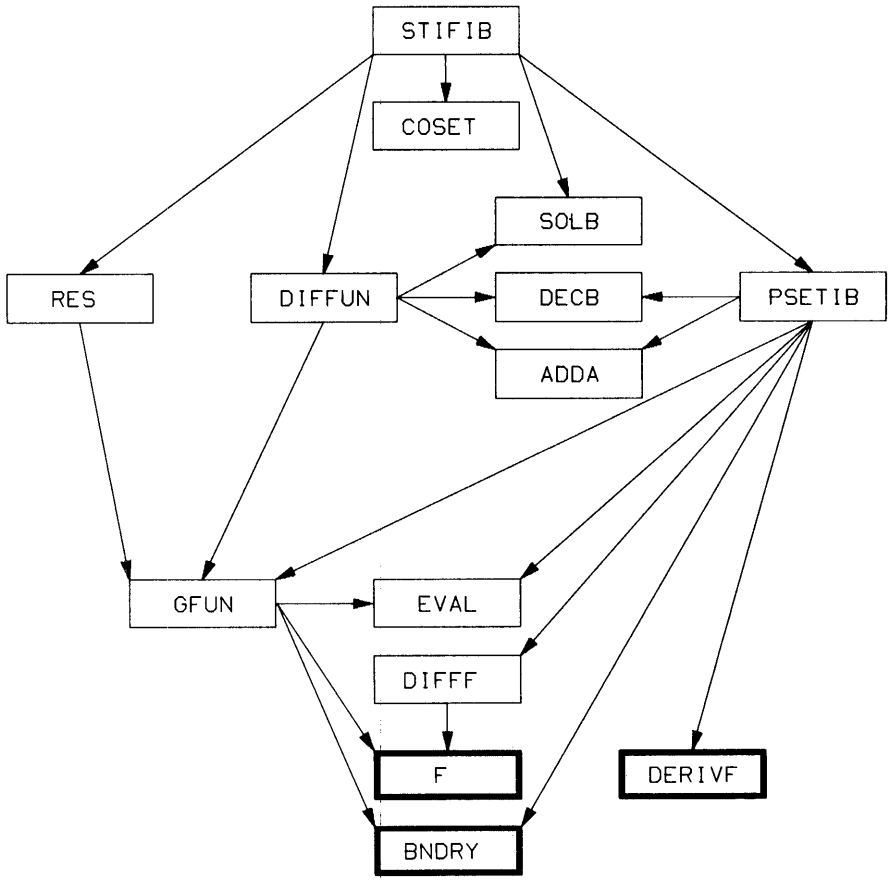


Fig. 2. Lines of communication between user routines and PDECOL package routines: Part 2, user routines are indicated by heavy borders

required by the package are allocated and defined in PDECOL. On the first call to the package, this routine checks the user's input for legality and performs initialization tasks such as setting problem parameters and calling the package routine INITAL. After the initialization is complete, PDECOL makes repeated calls to the core integrator routine STIFIB in order to advance the time variable a single step at a time until the user's desired output time is reached. At this time (since output times are not usually hit exactly) the package routine INTERP is called to interpolate the computed basis function coefficients to the desired output time. PDECOL also checks for possible error returns from STIFIB and either attempts to recover or provides the user with appropriate flags and/or messages.

VALUES. This is the routine the user must call in order to obtain the values of his computed approximate solution and its derivatives when his desired output time TOUT has been reached. VALUES performs the same basic function as EVAL (see below) except that it can evaluate the piecewise polynomial solution at *any* given spatial point or points—rather than just at collocation points. Since the computed solution is actually a continuous function, the user may desire solution values at arbitrary spatial points in the domain of interest. VALUES is structured much like EVAL. The basis function values are generated by BSPLVD as needed, and the user has the option of obtaining values at any number of points with a single call to VALUES.

INITAL. This routine performs functions which are very important to the entire package. Its main purpose is to determine the initial condition data for the package. This initial data consists of the basis function coefficients which are chosen so that the initial piecewise polynomial ((4.1) at $t = t_0$) interpolates the user's initial condition functions (2.4) at the collocation points. This requires that the collocation points and all of the basis function data be generated. INITAL generates (or causes to be generated) and stores the breakpoint sequence, the collocation points (see COLPNT), and the B-spline basis function and derivative values at the collocation points [3]. The basis function values and derivatives at the collocation points are calculated only once and saved so as to make the package efficient (see EVAL below).

COLPNT. The collocation points which are required by the collocation discretization procedure are calculated in this routine. In general we choose the collocation points to be the points in the domain $[x_L, x_R]$ at which the B-spline basis functions attain their unique maxima [1]. For the special piecewise polynomial spaces of order *KORD* with *NCC* = 2, a second more desirable choice is available (normal default choice). In this case the collocation points are chosen to be the unique *KORD* − *NCC* Gauss-Legendre quadrature points in each subinterval $[XBKPT(i), XBKPT(i + 1)]$, for $i = 1, 2, \ldots, NINT$, plus the two endpoints x_L and x_R. This choice of points for these piecewise polynomial spaces will produce more accurate approximate solutions for certain classes of problems [4].

BSPLVD, BSPLVN, and INTERV. These subprograms are the basic routines which are used to generate the B-spline basis function values and their derivative values at any desired points. They were developed and fully documented by deBoor [3] and are essentially unchanged from his original implementation. They provide for the reliable evaluation of arbitrary B-spline basis function values for polynomial orders up to 20.

STIFIB. This routine is the core integrator for the package. Its basic task is to advance the time value by taking a single time step. The routines STIFIB, COSET, DIFFUN, RES, ADDA, PSETIB, INTERP, DECB, and SOLB actually constitute a slightly modified version of an ODE package called GEARIB which has been developed by Hindmarsh [6]. The package GEARIB is designed to solve initial-value ODE problems of the form

$$A \frac{d\mathbf{y}}{dt} = \mathbf{g}(t, \mathbf{y}), \tag{6.1}$$

where $A(t, \mathbf{y})$ is a banded matrix, $\mathbf{y}$ is a vector, and $\mathbf{g}$ is a vector valued function. The semidiscrete equations (4.5) obtained by the collocation procedure above are particular examples of such equations. The time integration algorithms that are implemented in STIFIB are quite complex and we will discuss only the more important features. The methods used to advance the time are based on multipoint formulas of the form

$$\mathbf{y}_n = \sum_{j=1}^{k_1} \alpha_j \mathbf{y}_{n-j} + \Delta t \sum_{j=0}^{k_2} \beta_j \mathbf{y}'_{n-j}, \tag{6.2}$$

where $\mathbf{y}_k$ is an approximation to the exact value at $t = t_k$, $\mathbf{y}'$ denotes $d\mathbf{y}/dt$, $\Delta t = t_n - t_{n-1}$, and α_j and β_j are the method coefficients. Since $\beta_0 \neq 0$ for the methods used, the time integration methods (6.2) are *implicit* and so a nonlinear system of equations must in general be solved for every time step taken.

Proper choices of the coefficients in (6.2) can produce the standard Crank-Nicolson and backward difference methods so these techniques are generalizations of the classical, often used time integration methods.

There are two basic types of time integration formulas used in STIFIB. The first type of formula (Adams' methods) of order q, $1 \leq q \leq 12$, is obtained from (6.2) by setting $k_1 = 1$, $k_2 = q - 1$, and $\alpha_1 = 1$. The second type of formula (backward differentiation methods) of order q, $1 \leq q \leq 5$, is obtained by setting $k_1 = q$ and $k_2 = 0$. The remaining unspecified coefficients may be found in the book by Gear [5]. In general we recommend the use of the second type of formulas because of their better stability properties.

The time integration techniques in PDECOL feature automatic time step size and time integration formula order selection so as to efficiently solve the problem and yet achieve a user specified time integration error level.

For more specific details of the techniques and procedures implemented in STIFIB, we refer the reader to the comments in the program and the GEARIB report [6].

GFUN. The primary function of the routine GFUN is to properly evaluate the function $\mathbf{g}(t, \mathbf{y})$ of (6.1) when provided with input values of t and $\mathbf{y}$. In the package the vector function $\mathbf{g}(t, \mathbf{y})$ actually consists of values at the collocation points of the right-hand-side function $\mathbf{f}$, which defines the original PDE problem (2.1). Since the quantities actually being computed by the package (and the arguments input to GFUN) are basis function coefficients, GFUN must use these coefficients and the user defined subprograms F and BNDRY to generate the proper output values of $\mathbf{g}(t, \mathbf{y})$. The basis function coefficients are converted to piecewise polynomial values by EVAL (see below) and these values are then used to call the user's routines F and BNDRY. This conversion and evaluation process is performed at each collocation point in succession with BNDRY being called only for the boundary points. The first and last block rows of the A matrix (6.1) are also updated by GFUN to properly account for the boundary conditions (2.1).

GFUN is the routine which actually implements the collocation discretization process. It serves as the basic interface (via the method of lines) between the users' routines F and BNDRY and an ODE integrator (in this case a modified GEARIB). It serves the same interface role as the routine PDEONE [12] in a similar finite difference—method of lines implementation.

EVAL. The purpose of EVAL is to evaluate the piecewise polynomial (4.1) and its first two derivatives at a given collocation point, when the current basis function coefficients are provided as input. Since at any given point, at most *KORD* basis functions are nonzero, this computation involves computing three vector inner products where the vectors are of length *KORD*. This computation is the "innermost loop" in the package, since EVAL is called just prior to each call to the user's routine F (or BNDRY). Originally, the basis function values were not stored and were repeatedly computed (by BSPLVD) as needed. However, the expense of this repeated computation was so great that overall run times

were decreased by factors of from five to ten when it was decided to store all of the basis function (and derivative) values at the collocation points (see INITAL). This gain in speed is, of course, at the expense of additional storage.

PSETIB. This routine is required by STIFIB to compute and process the matrix $A - h\beta_0 (\partial \mathbf{g}/\partial \mathbf{y})$ where A is the banded matrix in (6.1) and $\partial \mathbf{g}/\partial \mathbf{y}$ is the banded Jacobian matrix of the function $\mathbf{g}(t, \mathbf{y})$ in (6.1). If the values of the partial derivatives of the user's function $\mathbf{f}$ in (2.1) with respect to $\mathbf{u}$, $\mathbf{u}_x$, and $\mathbf{u}_{xx}$ are known, then the Jacobian matrix $\partial \mathbf{g}/\partial \mathbf{y}$ can easily be computed using the chain rule. Once PSETIB has formed the desired matrix, it then LU decomposes the matrix for use in STIFIB by calling the package routine DECB. There are two options available in PSETIB for generating the needed partial derivatives and the user makes the selection. One option requires that the user construct a routine DERIVF to provide the partial derivative information (see the Algorithm). In the second option approximate partial derivatives are generated internally in the package routine DIFFF by finite difference quotients obtained by calling the user's routine F. The DERIVF option is only slightly more efficient (at most 10 percent faster). However, it does have the advantage that more accurate partial derivative values will be obtained. Inaccurate Jacobian matrix values do not directly affect the accuracy of the computed approximate solution, but can cause longer running times by forcing the package to take smaller time steps and to generate Jacobian matrices more frequently. For most problems we have found the second option totally adequate and recommend its use.

7. Example Problems

To illustrate the use of PDECOL, we present a main program and the user written subroutines required to solve the following problem on the interval [0, 1]:

$$\frac{\partial u}{\partial t} = v^2 \frac{\partial^2 u}{\partial x} + 2v \frac{\partial v}{\partial x} \frac{\partial u}{\partial x} - uv - u^2 + 10,$$

$$\frac{\partial v}{\partial t} = u^2 \frac{\partial^2 v}{\partial x^2} + 2u \frac{\partial u}{\partial x} \frac{\partial v}{\partial x} + \frac{\partial^2 u}{\partial x^2} + uv - v^2,$$

with boundary conditions

$$u = \tfrac{1}{2}, \qquad v = \pi \qquad \text{at } x = 0,$$

$$\frac{\partial u}{\partial x} + \sin(uv) = \tfrac{1}{2}, \qquad \frac{\partial v}{\partial x} - \cos(uv) = 1 \qquad \text{at } x = 1$$

and initial conditions

$$u = \tfrac{1}{2}(x + 1), \qquad v = \pi \qquad \text{at } t = 0.$$

Note that the initial conditions are consistent with the boundary conditions as required by PDECOL.

Figure 3 shows a main program and the user written subroutines F, BNDRY, UINIT, and DERIVF for this problem. Subroutine DERIVF is an optional routine which needs to be provided only if $MF = 11$ or 21 (see internal computer documentation for additional details). We present it here for completeness.

In Figure 4 we present some numerical results for this problem. These results were obtained using the piecewise polynomial space defined by $KORD = 4$, $NCC = 2$, and $NINT = 30$ with equally spaced breakpoints. The reader is referred to the internal program documentation in PDECOL for the definition of the quantities that appear in the figure. Since we do not know the exact solution to this problem, we are not able to determine the error in the calculated solution.

We next consider a relatively simple PDE problem where we know the exact solution. The purpose of this problem is to show the use of higher order methods

```
      PROGRAM TEST(UGOUT,TAPE3=UGOUT)
      COMMON /ENDPT/ XLEFT
      COMMON /GEAR0/ DTUSED,NQ,NSTEPS,NF,NJ
      DIMENSION U(2,31),XBKPT(31),SCTCH(10),WORK(5000),IWORK(500)
C
C INITIALIZE PARAMETERS AND PDECOL CALLING ARGUMENTS
C
      NPDE = 2
      NINT = 30
      NPTS = NINT + 1
      KORD = 4
      NCC = 2
      T0 = 0.0
      TOUT = 1.E-3
      DT = 1.E-7
      EPS = 1.E-4
      MF = 21
      INDEX = 1
      IWORK(1) = 5000
      IWORK(2) = 500
      DX = 1.0 / FLOAT(NPTS-1)
      DO 10 I=1,NPTS
        XBKPT(I) = FLOAT(I-1) * DX
   10 CONTINUE
      XLEFT = XBKPT(1)
C
C CALL THE PACKAGE TO INTEGRATE TO TIME T = TOUT
C
   20 CALL PDECOL(T0,TOUT,DT,XBKPT,EPS,NINT,KORD,NCC,NPDE,MF,INDEX
     *            WORK,IWORK)
C
C CHECK FOR EXECUTION ERRORS
C
      IF ( INDEX .NE. 0 ) GO TO 70
C
C OUTPUT PERFORMANCE DATA AND COMPUTED SOLUTION VALUES
C
      WRITE(3,30) TOUT,DTUSED,NSTEPS
   30 FORMAT(//10X,3HT= ,E10.3,7H   DT= ,E10.3,15H   TOTAL STEPS=,I5)
      CALL VALUES(XBKPT,U,SCTCH,NPDE,NPTS,NPTS,0,WORK)
      DO 60 K=1,NPDE
        WRITE(3,40) K
   40   FORMAT(/10X,13HPDE COMPONENT,I2/)
        WRITE(3,50) (U(K,I),I=1,NPTS)
   50   FORMAT(10X,5E12.4)
   60 CONTINUE
C
C SET NEW OUTPUT TIME AND CONTINUE THE INTEGRATION IF TOUT .LT. 11.0
C OTHERWISE, TERMINATE THE PROBLEM
C
      TOUT = TOUT * 10.0
      IF ( TOUT .LT. 11.0 ) GO TO 20
   70 WRITE(3,80) INDEX
   80 FORMAT(10X,7HINDEX= ,I3)
      STOP
      END
```

```
      SUBROUTINE F(T,X,U,UX,UXX,FVAL,NPDE)
      DIMENSION U(NPDE), US(NPDE), UXX(NPDE), FVAL(NPDE)
        FVAL(1) = U(2)*U(2)*UXX(1) - U(1)*U(2) - U(1)**2 + 10.0
     *          + 2.0*U(2)*UX(2)*UX(1)
        FVAL(2) = U(1)*U(1)*UXX(2) + U(1)*U(2) - U(2)**2
     *          + UXX(1) + 2.0*U(1)*UX(1)*UX(2)
      RETURN
      END
```

Fig. 3. Main program and user written subroutines for the first example problem

```
      SUBROUTINE BNDRY(T,X,U,UX,DBDU,DBDUX,DZDT,NPDE)
      DIMENSION U(NPDE), UX(NPDE), DZDT(NPDE)
      DIMENSION DBDU(NPDE,NPDE), DBDUX(NPDE,NPDE)
      COMMON /ENDPT/ XLEFT
      IF ( X .NE. XLEFT ) GO TO 10
        DBDU(1,1) = 1.0
        DBDU(1,2) = 0.0
        DBDU(2,1) = 0.0
        DBDU(2,2) = 1.0
        DBDUX(1,1) = 0.0
        DBDUX(1,2) = 0.0
        DBDUX(2,1) = 0.0
        DBDUX(2,2) = 0.0
        DZDT(1) = 0.0
        DZDT(2) = 0.0
      RETURN
   10   DBDU(1,1) = U(2) * COS( U(1) * U(2) )
        DBDU(1,2) = U(1) * COS( U(1) * U(2) )
        DBDU(2,1) = U(2) * SIN( U(1) * U(2) )
        DBDU(2,2) = U(1) * SIN( U(1) * U(2) )
        DBDUX(1,1) = 1.0
        DBDUX(1,2) = 0.0
        DBDUX(2,1) = 0.0
        DBDUX(2,2) = 1.0
        DZDT(1) = 0.0
        DZDT(2) = 0.0
      RETURN
      END



      SUBROUTINE UINIT(X,U,NPDE)
      DIMENSION U(NPDE)
C
C SET INITIAL CONDITIONS.  NOTE THAT  PI = 4.0*ATAN(1.0)
C
      U(1) = 0.5 * ( X - 1.0 )
      U(2) = 4.0 * ATAN( 1.0 )

      RETURN
      END



      SUBROUTINE DERIVF(T,X,U,UX,UXX,DFDU,DFDUX,DFDUXX,NPDE)
      DIMENSION U(NPDE), UX(NPDE), UXX(NPDE)
      DIMENSION DFDU(NPDE,NPDE), DFDUX(NPDE,NPDE), DFDUXX(NPDE,NPDE)
        DFDU(1,1) = -U(2) - 2.0*U(1)
        DFDU(1,2) = 2.0*U(2)*UXX(1) - U(1) + 2.0*UX(2)*UX(1)
        DFDU(2,1) = 2.0*U(1)*UXX(2) + U(2) + 2.0*UX(1)*UX(2)
        DFDU(2,2) = U(1) - 2.0*U(2)
C
        DFDUX(1,1) = 2.0*U(2)*UX(2)
        DFDUX(1,2) = 2.0*U(2)*UX(1)
        DFDUX(2,1) = 2.0*U(1)*UX(2)
        DFDUX(2,2) = 2.0*U(1)*UX(1)
C
        DFDUXX(1,1) = U(2)*U(2)
        DFDUXX(1,2) = 0.0
        DFDUXX(2,1) = 1.0
        DFDUXX(2,2) = U(1)*U(1)
      RETURN
      END
```

Fig. 3. (Continued)

and the execution times on a CDC-7600 computer. The problem is of a simple diffusion type and is described by

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \pi^2 \sin \pi x,$$

$$u(0, x) = 1, \qquad u(t, 0) = u(t, 1) = 1$$

for $0 < x < 1$ and $t > 0$. The exact solution for this problem is easily seen to be

$$u(t, x) = 1 + (\sin \pi x)(1 - e^{-\pi^2 t}).$$

Table I shows the results obtained for this problem at $t = 0.1$ with $NCC = 2$ for several values of *KORD* and *NINT*. Equally spaced breakpoints and $MF = 21$ were used in all cases. The values of *EPS*, the time integration error tolerance, were determined *by trial and error* to be as large as possible without affecting the spatial errors. The initial *DT* was chosen to be equal to *EPS*/2. *NSTEPS* is the number of time integration steps. The time, *TIME*, is the computer time in seconds (CDC-7600) required to produce the solution. The error, *ERROR*, is the maximum absolute difference between the exact solution and the numerical solution. The calculated order of convergence is shown in the last column. The numbers in parentheses are the theoretical values.

The preceding two examples illustrate the use of PDECOL and demonstrate the qualitative behavior of the higher order collocation methods that are implemented in the package. To consider all combinations of PDE methods that are built into PDECOL is virtually impossible since there are potentially over 3000 different methods in the package. We refer the reader to several other papers [9, 10] for additional results on the use, testing, and evaluation of PDECOL to solve partial differential equations and two-point boundary value problems. Some comparisons of PDECOL with low order finite difference methods are contained

```
T=  1.000E-03   DT=  3.400E-04   TOTAL STEPS=   11

PDE COMPONENT 1
  5.0000E-01  5.1915E-01  5.3759E-01  5.5547E-01  5.7294E-01
  5.9012E-01  6.0709E-01  6.2393E-01  6.4067E-01  6.5736E-01
  6.7402E-01  6.9065E-01  7.0726E-01  7.2386E-01  7.4046E-01
  7.5706E-01  7.7365E-01  7.9024E-01  8.0683E-01  8.2342E-01
  8.4002E-01  8.5661E-01  8.7321E-01  8.8982E-01  9.0645E-01
  9.2309E-01  9.9377E-01  9.5651E-01  9.7332E-01  9.9026E-01
  1.0074E+00

PDE COMPONENT 2
  3.1416E+00  3.1335E+00  3.1331E+00  3.1332E+00  3.1334E+00
  3.1335E+00  3.1336E+00  3.1337E+00  3.1337E+00  3.1338E+00
  3.1338E+00  3.1339E+00  3.1340E+00  3.1340E+00  3.1341E+00
  3.1341E+00  3.1342E+00  3.1342E+00  3.1343E+00  3.1343E+00
  3.1344E+00  3.1344E+00  3.1345E+00  3.1345E+00  3.1346E+00
  3.1347E+00  3.1347E+00  3.1348E+00  3.1349E+00  3.1349E+00
  3.1350E+00


T=  1.000E-02   DT=  2.052E-03   TOTAL STEPS=   21

PDE COMPONENT 1
  5.0000E-01  5.2478E-01  5.4920E-01  5.7300E-01  5.9609E-01
  6.1848E-01  6.4020E-01  6.6132E-01  6.8189E-01  7.0198E-01
  7.2165E-01  7.4097E-01  7.5999E-01  7.7877E-01  7.9738E-01
  8.1588E-01  9.3433E-01  8.5280E-01  8.7135E-01  8.9007E-01
  9.0902E-01  9.2829E-01  9.4797E-01  9.6817E-01  9.8898E-01
  1.0105E+00  1.0330E+00  1.0564E+00  1.0810E+00  1.1070E+00
  1.1345E+00
```

Fig. 4. Numerical results for the first example problem (continued on p. 342)

```
PDECCOMPONENT 2
  3.1416E+00  3.0943E+00  3.0722E+00  3.0634E+00  3.0606E+00
  3.0604E+00  3.0611E+00  3.0620E+00  3.0630E+00  3.0640E+00
  3.0649E+00  3.0658E+00  3.0667E+00  3.0675E+00  3.0684E+00
  3.0692E+00  3.0700E+00  3.0708E+00  3.0716E+00  3.0724E+00
  3.0733E+00  3.0742E+00  3.0752E+00  3.0763E+00  3.0773E+00
  3.0785E+00  3.0798E+00  3.0811E+00  3.0825E+00  3.0842E+00
  3.0861E+00

T=  1.000E-01   DT=  7.857E-03   TOTAL STEPS=   52

PDE COMPONENT 1
  5.0000E-01  5.4709E-01  5.9690E-01  6.4852E-01  7.0117E-01
  7.5419E-01  8.0704E-01  8.5929E-01  9.1062E-01  9.6079E-01
  1.0096E+00  1.0570E+00  1.1028E+00  1.1471E+00  1.1897E+00
  1.2308E+00  1.2702E+00  1.3081E+00  1.3444E+00  1.3793E+00
  1.4127E+00  1.4447E+00  1.4753E+00  1.5047E+00  1.5328E+00
  1.5598E+00  1.5855E+00  1.6102E+00  1.6339E+00  1.6566E+00
  1.6783E+00

PDE COMPONENT 2
  3.1416E+00  3.0064E+00  2.9094E+00  2.8409E+00  2.7943E+00
  2.7643E+00  2.7475E+00  2.7409E+00  2.7428E+00  2.7514E+00
  2.7657E+00  2.7847E+00  2.8079E+00  2.8347E+00  2.8647E+00
  2.8977E+00  2.9332E+00  2.9713E+00  3.0117E+00  3.0543E+00
  3.0991E+00  3.1459E+00  3.1948E+00  3.2457E+00  3.2986E+00
  3.3535E+00  3.4104E+00  3.4693E+00  3.5303E+00  3.5933E+00
  3.6584E+00

T=  1.000E+00   DT=  3.566E-01   TOTAL STEPS=   76

PDE COMPONENT 1
  5.0000E-01  5.4699E-01  5.9679E-01  6.4846E-01  7.0117E-01
  7.5420E-01  8.0695E-01  8.5895E-01  9.0985E-01  9.5940E-01
  1.0074E+00  1.0538E+00  1.0985E+00  1.1414E+00  1.1826E+00
  1.2221E+00  1.2599E+00  1.2962E+00  1.3308E+00  1.3640E+00
  1.3957E+00  1.4261E+00  1.4551E+00  1.4830E+00  1.5096E+00
  1.5351E+00  1.5596E+00  1.5830E+00  1.6055E+00  1.6271E+00
  1.6479E+00

PDE COMPONENT 2
  3.1416E+00  3.0030E+00  2.9035E+00  2.8341E+00  2.7879E+00
  2.7598E+00  2.7460E+00  2.7435E+00  2.7500E+00  2.7638E+00
  2.7837E+00  2.8085E+00  2.8374E+00  2.8699E+00  2.9054E+00
  2.9436E+00  2.9842E+00  3.0270E+00  3.0718E+00  3.1184E+00
  3.1669E+00  3.2170E+00  3.2688E+00  3.3223E+00  3.3774E+00
  3.4341E+00  3.4925E+00  3.5525E+00  3.6143E+00  3.6778E+00
  3.7432E+00


T=  1.000E+01   DT=  3.566E+00   TOTAL STEPS=   81

PDE COMPONENT 1
  5.0000E-01  5.4699E-01  5.9679E-01  6.4846E-01  7.0117E-01
  7.5420E-01  8.0695E-01  8.5895E-01  9.0985E-01  9.5940E-01
  1.0074E+00  1.0538E+00  1.0985E+00  1.1414E+00  1.1826E+00
  1.2221E+00  1.2599E+00  1.2962E+00  1.3308E+00  1.3640E+00
  1.3957E+00  1.4261E+00  1.4551E+00  1.4830E+00  1.5096E+00
  1.5351E+00  1.5596E+00  1.5830E+00  1.6055E+00  1.6271E+00
  1.6479E+00

PDE COMPONENT 2
  3.1416E+00  3.0030E+00  2.9035E+00  2.8341E+00  2.7879E+00
  2.7598E+00  2.7460E+00  2.7434E+00  2.7500E+00  2.7638E+00
  2.7837E+00  2.8085E+00  2.8374E+00  2.8699E+00  2.9054E+00
  2.9436E+00  2.9842E+00  3.0270E+00  3.0718E+00  3.1184E+00
  3.1669E+00  3.2170E+00  3.2688E+00  3.3223E+00  3.3774E+00
  3.4341E+00  3.4925E+00  3.5525E+00  3.6143E+00  3.6778E+00
  3.7432E+00
INDEX=   0
```

Fig. 4. (Continued)

Table I. Results at t = 0.1 for the Second Example Problem with *NCC* = 2 for Several Values of *KORD* and *NINT*

KORD	*NINT*	*EPS*	*NSTEPS*	*TIME* (sec)	*ERROR*	Order of convergence
3	10	3×10^{-5}	15	3.33×10^{-2}	1.14×10^{-3}	
3	20	1×10^{-5}	19	7.56×10^{-2}	2.94×10^{-4}	1.96 (2)
4	8	1×10^{-6}	25	9.53×10^{-2}	2.19×10^{-5}	
4	16	1×10^{-7}	33	2.32×10^{-1}	1.53×10^{-6}	3.90 (4)
5	4	1×10^{-6}	25	8.44×10^{-2}	1.38×10^{-5}	
5	8	1×10^{-8}	44	2.50×10^{-1}	4.33×10^{-7}	5.00 (5)
6	2	1×10^{-6}	25	6.61×10^{-2}	1.85×10^{-5}	
6	4	1×10^{-8}	41	1.84×10^{-1}	4.13×10^{-7}	5.50 (6)

in [9]. Our basic feelings are that for *smooth* problems and *high* accuracy demands, the higher order collocation techniques will be significantly more efficient, that is, produce more accuracy per unit of run time.

Our experience in using PDECOL has led us to firmly believe that PDECOL is a versatile software package that can solve a broad class of nonlinear partial differential equations. Its higher order methods can produce extremely accurate solutions quite efficiently when compared to lower order methods. The package is quite portable.

REFERENCES

1. CURRY, H.B., AND SCHOENBERG, I.J. On Polya frequency functions IV: The fundamental spline functions and their limits. *J. Anal. Math. 17* (1966), 71–107.
2. DEBOOR, C. On calculating with *B*-splines. *J. Approx. Theory 6*, 1 (July 1972), 50–62.
3. DEBOOR, C. Package for calculating with B-splines. *SIAM J. Numer. Anal. 14*, 3 (June 1977), 441–472.
4. DEBOOR, C., AND SWARTZ, B. Collocation at Gaussian points. *SIAM J. Numer. Anal. 10*, 4 (Sept. 1973), 582–606.
5. GEAR, C.W. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, Englewood Cliffs, N.J., 1971.
6. HINDMARSH, A.C. Preliminary documentation of GEARIB: Solution of implicit systems of ordinary differential equations with banded Jacobian. Rep. UCID-30130, Lawrence Livermore Lab., Livermore, Calif., 1976.
7. MADSEN, N.K., AND SINCOVEC, R.F. The numerical method of lines for the solution of nonlinear partial differential equations. In *Computational Methods in Nonlinear Mechanics*, J.T. Oden et al., Eds., Texas Institute for Computational Mechanics, Austin, Tex., 1974, pp. 371–380.
8. MADSEN, N.K., AND SINCOVEC, R.F. PDEPACK: A new tool for simulation. Proc. 1975 Summer Computer Simulation Conf., San Francisco, Calif., July 1975, pp. 144–149. (© 1975 by the Simulation Councils, Inc., P.O. Box 2228, La Jolla, Calif. 92037.)
9. MADSEN, N.K., AND SINCOVEC, R.F. General software for partial differential equations. In *Numerical Methods for Differential Systems*, L. Lapidus and W.E. Schiesser, Eds., Academic Press, New York, 1976, pp. 229–242.
10. SINCOVEC, R.F. On the relative efficiency of higher order collocation methods for solving two-point boundary value problems. *SIAM J. Numer. Anal. 14*, 1 (March 1977), 112–123.
11. SINCOVEC, R.F., AND MADSEN, N.K. Software for nonlinear partial differential equations. *ACM Trans. Math. Software 1*, 3 (Sept. 1975), 232–260.
12. SINCOVEC, R.F., AND MADSEN, N.K. ALGORITHM 494: PDEONE, solutions of systems of partial differential equations. *ACM Trans. Math. Software 1*, 3 (Sept. 1975), 261–263.
13. SWARTZ, B.K., AND VARGA, R.S. Error bounds for spline and *L*-spline interpolation. *J. Approx. Theory 6*, 1 (July 1972), 6–49.

ALGORITHM

[Summary information and a part of the listing is printed here. To facilitate understanding of the partial listing given here, lines 6110–6180 of the complete listing have been substituted for lines 4050–4060 of the complete listing, which is available from the ACM Algorithms Distribution Service.]

NAME(n): indicates a Fortran module with n records
NAME^{D}(n): indicates "NAME" contains test data
NAME^{M}(n): indicates "NAME" contains machine dependent data

Contents: PDECOL (924), VALUES (70), BLOCKD^{M} (19), INITAL (84), COLPNT (222), BSPLVD (72), BSPLVN (46), INTERV (88), STIFIB (416), GFUN (70), EVAL (28), DIFFUN (28), ADDA (45), RES (53), PSETIB (102), DIFFF (46), INTERP (31), COSET (179), DECB (87), SOLB (47)

Test package: TEST1 (56), F1 (8), BNDRY1 (28), UINITI (9), DERIVF1 (19), TEST2 (61), F2 (6), BNDRY2 (6), UNINIT2 (5), DERIVF2 (6), TRUSOL2 (5), TSIN2^{D} (21), TEST3 (61), F3 (10), BNDRY3 (27), UINIT3 (5), DERIVF3 (17), TRUSOL3 (6), TSIN3^{D} (25), TEST4 (64), F4 (5), BNDRY4 (12), UINIT4 (5), DERIVF4 (6), TRUSOL4 (11), TSIN4^{D} (19), TEST5 (61), F5 (5), BNDRY5 (13), UINIT5 (5), DERIVF5 (6), TRUSOL5 (5), TSIN5^{D} (24)

```
      SUBROUTINE PDECOL(TØ,TOUT,DT,XBKPT,EPS,NINT,KORD,NCC,NPDE,MF,          1Ø
     *                  INDEX,WORK,IWORK)                                    2Ø
C                                                                           3Ø
C-----------------------------------------------------------------------    4Ø
C-----------------------------------------------------------------------    5Ø
C                                                                           6Ø
C THIS IS THE MARCH 24, 1978 VERSION OF PDECOL.                             7Ø
C                                                                           8Ø
C THIS PACKAGE WAS CONSTRUCTED SO AS TO CONFORM TO AS MANY ANSI-FORTRAN     9Ø
C RULES AS WAS CONVENIENTLY POSSIBLE.  THE FORTRAN USED VIOLATES ANSI       1ØØ
C STANDARDS IN THE TWO WAYS LISTED BELOW....                                11Ø
C                                                                           12Ø
C   1. SUBSCRIPTS OF THE GENERAL FORM C*V1 + V2 + V3 ARE USED               13Ø
C      (POSSIBLY IN A PERMUTED ORDER), WHERE C IS AN INTEGER CONSTANT       14Ø
C      AND V1, V2, AND V3 ARE INTEGER VARIABLES.                            15Ø
C                                                                           16Ø
C   2. ARRAY NAMES APPEAR SINGLY IN DATA STATEMENTS IN THE ROUTINES         17Ø
C      BSPLVN AND COSET.                                                    18Ø
C                                                                           19Ø
C MACHINE DEPENDENT FEATURES......                                          2ØØ
C                                                                           21Ø
C THIS VERSION OF PDECOL WAS DESIGNED FOR USE ON CDC MACHINES WITH          22Ø
C A WORD LENGTH OF 6Ø BITS.  WE DO NOT RECOMMEND THE USE OF PDECOL WITH     23Ø
C WORD LENGTHS OF LESS THAN 48 BITS.  THE MOST IMPORTANT MACHINE            24Ø
C AND WORD LENGTH DEPENDENT CONSTANTS ARE DEFINED IN THE BLOCK DATA         25Ø
C AND IN SUBROUTINES COLPNT AND COSET.  THE USER SHOULD CHECK THESE         26Ø
C CAREFULLY FOR APPROPRIATENESS FOR HIS LOCAL SITUATION.  THE FORTRAN       27Ø
C FUNCTIONS USED BY EACH ROUTINE ARE LISTED IN THE COMMENTS TO              28Ø
C FACILITATE CONVERSION TO DOUBLE PRECISION.                                29Ø
C                                                                           3ØØ
C-----------------------------------------------------------------------    31Ø
C-----------------------------------------------------------------------    32Ø
C                                                                           33Ø
C-----------------------------------------------------------------------    34Ø
C PDECOL IS THE DRIVER ROUTINE FOR A SOPHISTICATED PACKAGE OF               35Ø
C SUBROUTINES WHICH IS DESIGNED TO SOLVE THE GENERAL SYSTEM OF              36Ø
C NPDE NONLINEAR PARTIAL DIFFERENTIAL EQUATIONS OF AT MOST SECOND           37Ø
C ORDER ON THE INTERVAL (XLEFT,XRIGHT) FOR T .GT. TØ WHICH IS OF THE        38Ø
C FORM....                                                                  39Ø
C                                                                           4ØØ
C      DU/DT  =  F( T, X, U, UX, UXX )                                     41Ø
C                                                                           42Ø
C WHERE                                                                     43Ø
C                                                                           44Ø
C          U  =  ( U(1),  U(2), ... ,  U(NPDE) )                           45Ø
C         UX  =  ( UX(1), UX(2), ... , UX(NPDE) )                          46Ø
C        UXX  =  (UXX(1),UXX(2), ... ,UXX(NPDE) ) .                        47Ø
C                                                                           48Ø
C EACH U(K) IS A FUNCTION OF THE SCALAR QUANTITIES T AND X.                 49Ø
C UX(K) REPRESENTS THE FIRST PARTIAL DERIVATIVE OF U(K) WITH RESPECT        5ØØ
C TO THE VARIABLE X,  UXX(K) REPRESENTS THE SECOND PARTIAL DERIVATIVE       51Ø
C OF U(K) WITH RESPECT TO THE VARIABLE X, AND DU/DT IS THE VECTOR OF        52Ø
C PARTIAL DERIVATIVES OF U WITH RESPECT TO THE TIME VARIABLE T.             53Ø
C F  REPRESENTS AN ARBITRARY VECTOR VALUED FUNCTION WHOSE NPDE              54Ø
C COMPONENTS DEFINE THE RESPECTIVE PARTIAL DIFFERENTIAL EQUATIONS OF        55Ø
C THE PDE SYSTEM.  SEE SUBROUTINE F DESCRIPTION BELOW.                      56Ø
C                                                                           57Ø
C BOUNDARY CONDITIONS                                                       58Ø
C                                                                           59Ø
```

```
C   DEPENDING ON THE TYPE OF PDE(S), Ø, 1, OR 2 BOUNDARY CONDITIONS          6ØØ
C   ARE REQUIRED FOR EACH PDE IN THE SYSTEM. THESE ARE IMPOSED AT XLEFT      61Ø
C   AND/ØR XRIGHT AND EACH MUST BE OF THE FORM....                           62Ø
C                                                                            63Ø
C        B(U,UX)  =  Z(T)                                                    64Ø
C                                                                            65Ø
C   WHERE  B  AND  Z  ARE ARBITRARY VECTOR VALUED FUNCTIONS WITH             66Ø
C   NPDE COMPONENTS AND  U, UX, AND T  ARE AS ABOVE.  THESE BOUNDARY         67Ø
C   CONDITIONS MUST BE CONSISTENT WITH THE INITIAL CONDITIONS WHICH ARE      68Ø
C   DESCRIBED NEXT.                                                          69Ø
C                                                                            7ØØ
C INITIAL CONDITIONS                                                         71Ø
C                                                                            72Ø
C   EACH SOLUTION COMPONENT  U(K)  IS ASSUMED TO BE A KNOWN (USER            73Ø
C   PROVIDED) FUNCTION OF  X  AT THE INITIAL TIME T = TØ.  THE               74Ø
C   INITIAL CONDITION FUNCTIONS MUST BE CONSISTENT WITH THE BOUNDARY         75Ø
C   CONDITIONS ABOVE, I.E. THE INITIAL CONDITION FUNCTIONS MUST              76Ø
C   SATISFY THE BOUNDARY CONDITIONS FOR T = TØ.  SEE SUBROUTINE UINIT        77Ø
C   DESCRIPTION BELOW.                                                       78Ø
C-------------------------------------------------------------------------  79Ø
C                                                                            8ØØ
C REQUIRED USER SUPPLIED SUBROUTINES                                         81Ø
C                                                                            82Ø
C THE USER IS REQUIRED TO CONSTRUCT THREE SUBPROGRAMS AND A MAIN             83Ø
C PROGRAM WHICH DEFINE THE PDE PROBLEM WHOSE SOLUTION IS TO BE               84Ø
C ATTEMPTED.  THE THREE SUBPROGRAMS ARE...                                   85Ø
C                                                                            86Ø
C 1)  SUBROUTINE F( T, X, U, UX, UXX, FVAL, NPDE )                          87Ø
C     DIMENSION U(NPDE), UX(NPDE), UXX(NPDE), FVAL(NPDE)                     88Ø
C        THIS ROUTINE DEFINES THE DESIRED PARTIAL DIFFERENTIAL               89Ø
C        EQUATIONS TO BE SOLVED.  THE PACKAGE PROVIDES VALUES OF THE         9ØØ
C        INPUT SCALARS T AND X AND INPUT ARRAYS (LENGTH NPDE) U, UX,         91Ø
C        AND UXX, AND THE USER MUST CONSTRUCT THIS ROUTINE TO COMPUTE        92Ø
C        THE OUTPUT ARRAY FVAL (LENGTH NPDE) WHICH CONTAINS THE              93Ø
C        CORRESPONDING VALUES OF THE RIGHT HAND SIDES OF THE DESIRED         94Ø
C        PARTIAL DIFFERENTIAL EQUATIONS, I.E.                                95Ø
C                                                                            96Ø
C        FVAL(K) = THE VALUE OF THE RIGHT HAND SIDE OF THE K-TH PDE IN       97Ø
C                  THE PDE SYSTEM ABOVE, FOR K = 1 TO NPDE.                  98Ø
C                                                                            99Ø
C        THE INCOMING VALUE OF THE SCALAR QUANTITY X WILL BE A              1ØØØ
C        COLLOCATION POINT VALUE (SEE INITAL AND COLPNT) AND THE            1Ø1Ø
C        INCOMING VALUES IN THE ARRAYS U, UX AND UXX CORRESPOND TO THIS     1Ø2Ø
C        POINT X AND TIME T.                                                1Ø3Ø
C     RETURN                                                                1Ø4Ø
C     END                                                                   1Ø5Ø
C                                                                           1Ø6Ø
C 2)  SUBROUTINE BNDRY( T, X, U, UX, DBDU, DBDUX, DZDT, NPDE )             1Ø7Ø
C     DIMENSION U(NPDE), UX(NPDE), DZDT(NPDE)                               1Ø8Ø
C     DIMENSION DBDU(NPDE,NPDE), DBDUX(NPDE,NPDE)                           1Ø9Ø
C        THIS ROUTINE IS USED TO PROVIDE THE PDE PACKAGE WITH NEEDED        11ØØ
C        INFORMATION ABOUT THE BOUNDARY CONDITION FUNCTIONS B AND Z         111Ø
C        ABOVE.  THE PACKAGE PROVIDES VALUES OF THE INPUT VARIABLES         112Ø
C        T, X, U, AND UX, AND THE USER IS TO DEFINE THE CORRESPONDING       113Ø
C        OUTPUT VALUES OF THE DERIVATIVES OF THE FUNCTIONS B AND Z          114Ø
C        WHERE....                                                          115Ø
C           DBDU(K,J) = PARTIAL DERIVATIVE OF THE K-TH COMPONENT OF THE     116Ø
C                       VECTOR FUNCTION B(U,UX) ABOVE WITH RESPECT TO       117Ø
C                       THE J-TH VARIABLE U(J).                             118Ø
C         DBDUX(K,J) = PARTIAL DERIVATIVE OF THE K-TH COMPONENT OF THE      119Ø
C                       VECTOR FUNCTION B(U,UX) ABOVE WITH RESPECT TO       12ØØ
C                       THE J-TH VARIABLE UX(J).                            121Ø
C             DZDT(K) = DERIVATIVE OF THE K-TH COMPONENT OF THE VECTOR      122Ø
C                       FUNCTION Z(T) ABOVE WITH RESPECT TO THE             123Ø
C                       VARIABLE T.                                         124Ø
C        NOTE... THE INCOMING VALUE OF X WILL BE EITHER XLEFT OR XRIGHT.    125Ø
C        IF NO BOUNDARY CONDITION IS DESIRED FOR SAY THE K-TH PDE AT        126Ø
C        ONE OR BOTH OF THE ENDPOINTS XLEFT OR XRIGHT, THEN DBDU(K,K)       127Ø
C        AND DBDUX(K,K) SHOULD BOTH BE SET TO ZERO WHEN BNDRY IS            128Ø
C        CALLED FOR THAT POINT.  WE REFER TO THIS AS A NULL BOUNDARY        129Ø
C        CONDITION.  THIS ROUTINE CAN BE STRUCTURED AS FOLLOWS...           13ØØ
C        THE COMMON BLOCK /ENDPT/ IS NOT A PART OF PDECOL AND               131Ø
C        MUST BE SUPPLIED AND DEFINED BY THE USER.                          132Ø
C     COMMON /ENDPT/ XLEFT                                                  133Ø
C     IF( X .NE. XLEFT ) GO TO 1Ø                                           134Ø
C        HERE DEFINE AND SET PROPER VALUES FOR DBDU(K,J), DBDUX(K,J),       135Ø
```

```
C        AND DZDT(K) FOR K,J = 1 TO NPDE FOR THE LEFT BOUNDARY POINT      136Ø
C        X = XLEFT.                                                       137Ø
C     RETURN                                                              138Ø
C  1Ø CONTINUE                                                           139Ø
C        HERE DEFINE AND SET PROPER VALUES FOR DBDU(K,J), DBDUX(K,J),     14ØØ
C        AND DZDT(K) FOR K,J = 1 TO NPDE FOR THE RIGHT BOUNDARY POINT     141Ø
C        X = XRIGHT.                                                      142Ø
C     RETURN                                                              143Ø
C     END                                                                 144Ø
C                                                                         145Ø
C 3)  SUBROUTINE UINIT( X, U, NPDE )                                     146Ø
C     DIMENSION U(NPDE)                                                   147Ø
C        THIS ROUTINE IS USED TO PROVIDE THE PDE PACKAGE WITH THE         148Ø
C        NEEDED INITIAL CONDITION FUNCTION VALUES.  THE PACKAGE           149Ø
C        PROVIDES A VALUE OF THE INPUT VARIABLE X, AND THE USER IS TO     15ØØ
C        DEFINE THE PROPER INITIAL VALUES (AT T = TØ) FOR ALL OF THE      151Ø
C        PDE COMPONENTS, I.E.                                             152Ø
C           U(K) = DESIRED INITIAL VALUE OF PDE COMPONENT U(K) AT         153Ø
C                  X AND T = TØ FOR K = 1 TO NPDE.                        154Ø
C        NOTE... THE INCOMING VALUE OF X WILL BE A COLLOCATION POINT      155Ø
C        VALUE.  THE INITIAL CONDITIONS AND BOUNDARY CONDITIONS           156Ø
C        MUST BE CONSISTENT (SEE ABOVE).                                  157Ø
C     RETURN                                                              158Ø
C     END                                                                 159Ø
C-------------------------------------------------------------------------16ØØ
C                                                                         161Ø
C OPTIONAL USER SUPPLIED SUBROUTINE                                       162Ø
C                                                                         163Ø
C IF THE USER DESIRES TO USE THE MF = 11 OR 21 OPTION IN ORDER TO SAVE    164Ø
C ABOUT 1Ø-2Ø PERCENT IN EXECUTION TIME (SEE BELOW), THEN THE USER MUST   165Ø
C PROVIDE THE FOLLOWING SUBROUTINE WHICH PROVIDES INFORMATION ABOUT THE   166Ø
C DERIVATIVES OF THE FUNCTION F ABOVE. THIS PROVIDES FOR MORE EFFICIENT   167Ø
C JACOBIAN MATRIX GENERATION.  ON MOST COMPUTER SYSTEMS, THE USER WILL    168Ø
C BE REQUIRED TO SUPPLY THIS SUBROUTINE AS A DUMMY SUBROUTINE IF THE      169Ø
C OPTIONS MF = 12 OR 22 ARE USED (SEE BELOW).                             17ØØ
C                                                                         171Ø
C 1)  SUBROUTINE DERIVF( T, X, U, UX, UXX, DFDU, DFDUX, DFDUXX, NPDE )    172Ø
C     DIMENSION U(NPDE), UX(NPDE), UXX(NPDE)                              173Ø
C     DIMENSION DFDU(NPDE,NPDE), DFDUX(NPDE,NPDE), DFDUXX(NPDE,NPDE)      174Ø
C        THE PACKAGE PROVIDES VALUES OF THE INPUT VARIABLES T, X, U, UX,  175Ø
C        AND UXX, AND THE USER SHOULD CONSTRUCT THIS ROUTINE TO PROVIDE   176Ø
C        THE FOLLOWING CORRESPONDING VALUES OF THE OUTPUT ARRAYS          177Ø
C        DFDU, DFDUX, AND DFDUXX FOR K,J = 1 TO NPDE...                   178Ø
C           DFDU(K,J) = PARTIAL DERIVATIVE OF THE K-TH COMPONENT OF THE   179Ø
C                       PDE DEFINING FUNCTION  F  WITH RESPECT TO THE     18ØØ
C                       VARIABLE U(J).                                    181Ø
C          DFDUX(K,J) = PARTIAL DERIVATIVE OF THE K-TH COMPONENT OF THE   182Ø
C                       PDE DEFINING FUNCTION  F  WITH RESPECT TO THE     183Ø
C                       VARIABLE UX(J).                                   184Ø
C         DFDUXX(K,J) = PARTIAL DERIVATIVE OF THE K-TH COMPONENT OF THE   185Ø
C                       PDE DEFINING FUNCTION  F  WITH RESPECT TO THE     186Ø
C                       VARIABLE UXX(J).                                  187Ø
C        NOTE... THE INCOMING VALUE OF  X  WILL BE A COLLOCATION POINT     188Ø
C        VALUE.                                                           189Ø
C     RETURN                                                              19ØØ
C     END                                                                 191Ø
C                                                                         271Ø
C USE OF PDECOL                                                           272Ø
C                                                                         273Ø
C PDECOL IS CALLED ONCE FOR EACH DESIRED OUTPUT VALUE (TOUT) OF THE       274Ø
C TIME T, AND IT IN TURN MAKES REPEATED CALLS TO THE CORE INTEGRATOR,     275Ø
C STIFIB, WHICH ADVANCES THE TIME BY TAKING SINGLE STEPS UNTIL            276Ø
C T .GE. TOUT.  INTERPOLATION TO THE EXACT TIME TOUT IS THEN DONE.        277Ø
C SEE TOUT BELOW.                                                         278Ø
C                                                                         279Ø
C                                                                         28ØØ
C SUMMARY OF SUGGESTED INPUT VALUES                                       281Ø
C                                                                         282Ø
C   IT IS OF COURSE IMPOSSIBLE TO SUGGEST INPUT PARAMETER VALUES WHICH    283Ø
C   ARE APPROPRIATE FOR ALL PROBLEMS.  THE FOLLOWING SUGGESTIONS ARE TO   284Ø
C   BE USED ONLY IF YOU HAVE NO IDEA OF BETTER VALUES FOR YOUR PROBLEM.   285Ø
C                                                                         286Ø
C   DT    = 1.E-1Ø                                                        287Ø
C   XBKPT = CHOOSE  NINT+1  EQUALLY SPACED VALUES SUCH THAT XBKPT(1) =    288Ø
C           XLEFT AND  XBKPT(NINT+1) = XRIGHT.                            289Ø
C   EPS   = 1.E-4                                                         29ØØ
```

```
C   NINT  = ENOUGH SO THAT ANY FINE STRUCTURE OF THE PROBLEM MAY BE        291Ø
C           RESOLVED.                                                     292Ø
C   KORD  = 4                                                             293Ø
C   NCC   = 2                                                             294Ø
C   MF    = 22                                                            295Ø
C   INDEX = 1 (ON FIRST CALL ONLY, THEN Ø THEREAFTER).                    296Ø
C                                                                         297Ø
C                                                                         298Ø
C THE INPUT PARAMETERS ARE..                                              299Ø
C   TØ    =  THE INITIAL VALUE OF T, THE INDEPENDENT VARIABLE             3ØØØ
C              (USED ONLY ON FIRST CALL).                                 3Ø1Ø
C   TOUT  =  THE VALUE OF T AT WHICH OUTPUT IS DESIRED NEXT.  SINCE       3Ø2Ø
C              THE PACKAGE CHOOSES ITS OWN TIME STEP SIZES, THE           3Ø3Ø
C              INTEGRATION WILL NORMALLY GO SLIGHTLY BEYOND TOUT          3Ø4Ø
C              AND THE PACKAGE WILL INTERPOLATE TO T = TOUT.              3Ø5Ø
C   DT    =  THE INITIAL STEP SIZE IN T, IF INDEX = 1, OR, THE            3Ø6Ø
C              MAXIMUM STEP SIZE ALLOWED (MUST BE .GT. Ø), IF INDEX = 3.  3Ø7Ø
C              USED FOR INPUT ONLY WHEN INDEX = 1 OR 3. SEE BELOW.        3Ø8Ø
C   XBKPT =  THE ARRAY OF PIECEWISE POLYNOMIAL BREAKPOINTS.               3Ø9Ø
C              THE NINT+1 VALUES MUST BE STRICTLY INCREASING WITH         31ØØ
C              XBKPT(1) = XLEFT AND XBKPT(NINT+1) = XRIGHT (USED ONLY     311Ø
C              ON FIRST CALL).                                            312Ø
C   EPS   =  THE RELATIVE TIME ERROR BOUND  (USED ONLY ON THE             313Ø
C              FIRST CALL, UNLESS INDEX = 4).  SINGLE STEP ERROR          314Ø
C              ESTIMATES DIVIDED BY CMAX(I) WILL BE KEPT LESS THAN        315Ø
C              EPS IN ROOT-MEAN-SQUARE NORM. THE VECTOR CMAX OF WEIGHTS   316Ø
C              IS COMPUTED IN PDECOL.  INITIALLY CMAX(I) IS SET TO        317Ø
C              ABS(C(I)), WITH A DEFAULT VALUE OF 1 IF ABS(C(I)) .LT. 1.  318Ø
C              THEREAFTER, CMAX(I) IS THE LARGEST VALUE                   319Ø
C              OF ABS(C(I)) SEEN SO FAR, OR THE INITIAL CMAX(I) IF        32ØØ
C              THAT IS LARGER.  TO ALTER EITHER OF THESE, CHANGE THE      321Ø
C              APPROPRIATE STATEMENTS IN THE DO-LOOPS ENDING AT           322Ø
C              STATEMENTS 5Ø AND 13Ø BELOW.  THE USER SHOULD EXERCISE     323Ø
C              SOME DISCRETION IN CHOOSING EPS.  IN GENERAL, THE          324Ø
C              OVERALL RUNNING TIME FOR A PROBLEM WILL BE GREATER IF      325Ø
C              EPS IS CHOSEN SMALLER. THERE IS USUALLY LITTLE REASON TO   326Ø
C              CHOOSE EPS MUCH SMALLER THAN THE ERRORS WHICH ARE BEING    327Ø
C              INTRODUCED BY THE USERS CHOICE OF THE POLYNOMIAL SPACE.    328Ø
C              SEE RELATED COMMENTS CONCERNING CMAX BELOW STATEMENT 4Ø.   329Ø
C   NINT  =  THE NUMBER OF SUBINTERVALS INTO WHICH THE SPATIAL DOMAIN     33ØØ
C              (XLEFT,XRIGHT) IS TO BE DIVIDED (MUST BE .GE. 1)           331Ø
C              (USED ONLY ON FIRST CALL).                                 332Ø
C   KORD  =  THE ORDER OF THE PIECEWISE POLYNOMIAL SPACE TO BE USED.      333Ø
C              ITS VALUE MUST BE GREATER THAN 2 AND LESS THAN 21.  FOR    334Ø
C              FIRST ATTEMPTS WE RECOMMEND KORD = 4.  IF THE SOLUTION     335Ø
C              IS SMOOTH AND MUCH ACCURACY IS DESIRED, HIGHER VALUES      336Ø
C              MAY PROVE TO BE MORE EFFICIENT.  WE HAVE SELDOM USED       337Ø
C              VALUES OF KORD IN EXCESS OF 8 OR 9, THOUGH THEY ARE        338Ø
C              AVAILABLE FOR USE IN PDECOL (USED ONLY ON FIRST CALL).     339Ø
C   NCC   =  THE NUMBER OF CONTINUITY CONDITIONS TO BE IMPOSED ON THE     34ØØ
C              APPROXIMATE SOLUTION AT THE BREAKPOINTS IN XBKPT.          341Ø
C              NCC MUST BE GREATER THAN 1 AND LESS THAN KORD.  WE         342Ø
C              RECOMMEND THE USE OF NCC = 2                               343Ø
C              SINCE THEORY PREDICTS THAT DRAMATICALLY MORE               344Ø
C              ACCURATE RESULTS CAN OFTEN BE OBTAINED USING THIS CHOICE   345Ø
C              (USED ONLY ON FIRST CALL).                                 346Ø
C   NPDE  =  THE NUMBER OF PARTIAL DIFFERENTIAL EQUATIONS IN THE SYSTEM   347Ø
C              TO BE SOLVED (USED ONLY ON FIRST CALL).                    348Ø
C   MF    =  THE METHOD FLAG  (USED ONLY ON FIRST CALL, UNLESS           349Ø
C              INDEX = 4).  ALLOWED VALUES ARE 11, 12, 21, 22.            35ØØ
C              FOR FIRST ATTEMPTS WE RECOMMEND THE USE OF MF = 22.        351Ø
C              MF HAS TWO DECIMAL DIGITS, METH AND MITER                  352Ø
C              (MF = 1Ø*METH + MITER).                                    353Ø
C              METH IS THE BASIC METHOD INDICATOR..                       354Ø
C                METH = 1  MEANS THE ADAMS METHODS (GENERALIZATIONS OF    355Ø
C                          CRANK-NICOLSON).                               356Ø
C                METH = 2  MEANS THE BACKWARD DIFFERENTIATION             357Ø
C                          FORMULAS (BDF), OR STIFF METHODS OF GEAR.      358Ø
C              MITER IS THE ITERATION METHOD INDICATOR                    359Ø
C              AND DETERMINES HOW THE JACOBIAN MATRIX IS                  36ØØ
C              TO BE COMPUTED..                                           361Ø
C                MITER = 1 MEANS CHORD METHOD WITH ANALYTIC JACOBIAN.     362Ø
C                          FOR THIS USER SUPPLIES SUBROUTINE DERIVF.      363Ø
C                          SEE DESCRIPTION ABOVE.                         364Ø
C                MITER = 2 MEANS CHORD METHOD WITH JACOBIAN CALCULATED    365Ø
C                          INTERNALLY BY FINITE DIFFERENCES.  SEE         366Ø
```

```
C                          SUBROUTINES PSETIB AND DIFFF.                    367Ø
C   INDEX =  INTEGER USED ON INPUT TO INDICATE TYPE OF CALL,                368Ø
C              WITH THE FOLLOWING VALUES AND MEANINGS..                     369Ø
C                 1    THIS IS THE FIRST CALL FOR THIS PROBLEM.             37ØØ
C                 Ø    THIS IS NOT THE FIRST CALL FOR THIS PROBLEM,         371Ø
C                      AND INTEGRATION IS TO CONTINUE.                      372Ø
C                 2    SAME AS Ø EXCEPT THAT TOUT IS TO BE HIT              373Ø
C                      EXACTLY (NO INTERPOLATION IS DONE).  SEE NOTE        374Ø
C                      BELOW.  ASSUMES TOUT .GE. THE CURRENT T.             375Ø
C                      IF TOUT IS .LT. THE CURRENT TIME, THEN TOUT IS       376Ø
C                      RESET TO THE CURRENT TIME AND CONTROL IS             377Ø
C                      RETURNED TO THE USER.  A CALL TO VALUES WILL         378Ø
C                      PRODUCE ANSWERS FOR THE NEW VALUE OF TOUT.           379Ø
C                 3    SAME AS Ø EXCEPT CONTROL RETURNS TO CALLING          38ØØ
C                      PROGRAM AFTER ONE STEP.  TOUT IS IGNORED AND         381Ø
C                      DT MUST BE SET .GT. Ø TO A MAXIMUM ALLOWED           382Ø
C                      DT VALUE. SEE ABOVE.                                 383Ø
C                 4    THIS IS NOT THE FIRST CALL FOR THE PROBLEM,          384Ø
C                      AND THE USER HAS RESET EPS AND/OR MF.                385Ø
C              SINCE THE NORMAL OUTPUT VALUE OF INDEX IS Ø,                 386Ø
C              IT NEED NOT BE RESET FOR NORMAL CONTINUATION.                387Ø
C                                                                           388Ø
C NOTE.. THE PACKAGE MUST HAVE TAKEN AT LEAST ONE SUCCESSFUL TIME           389Ø
C STEP BEFORE A CALL WITH INDEX = 2 OR 4 IS ALLOWED.                        39ØØ
C AFTER THE INITIAL CALL, IF A NORMAL RETURN OCCURRED AND A NORMAL          391Ø
C CONTINUATION IS DESIRED, SIMPLY RESET TOUT AND CALL AGAIN.                392Ø
C ALL OTHER PARAMETERS WILL BE READY FOR THE NEXT CALL.                     393Ø
C A CHANGE OF PARAMETERS WITH INDEX = 4 CAN BE MADE AFTER                   394Ø
C EITHER A SUCCESSFUL OR AN UNSUCCESSFUL RETURN PROVIDED AT LEAST           395Ø
C ONE SUCCESSFUL TIME STEP HAS BEEN MADE.                                   396Ø
C                                                                           397Ø
C   WORK  =  FLOATING POINT WORKING ARRAY FOR PDECOL.  WE RECOMMEND         398Ø
C              THAT IT BE INITIALIZED TO ZERO BEFORE THE FIRST CALL         399Ø
C              TO PDECOL.  ITS TOTAL LENGTH MUST BE AT LEAST                4ØØØ
C                                                                           4Ø1Ø
C              KORD + 4*NPDE + 9*NPDE**2 + NCPTS*(3*KCRD + 2) +             4Ø2Ø
C              NPDE*NCPTS*(3*ML + MAXDER + 7)                               4Ø3Ø
C                                                                           4Ø4Ø
C WHERE...                                                                  611Ø
C                                                                           612Ø
C      NCPTS = KORD*NINT - NCC*(NINT-1)                                     613Ø
C         ML = NPDE*(KORD+IQUAD-1) - 1                                      614Ø
C      IQUAD = 1 IF KORD = 3 AND A NULL BOUNDARY CONDITION EXISTS           615Ø
C      IQUAD = Ø OTHERWISE                                                  616Ø
C     MAXDER = 5 UNLESS OTHERWISE SET BY THE USER INTO /OPTION/.            617Ø
C                                                                           618Ø
C   IWORK =  INTEGER WORKING ARRAY FOR PDECOL.  THE FIRST TWO               4Ø7Ø
C              LOCATIONS MUST BE DEFINED AS FOLLOWS...                      4Ø8Ø
C              IWORK(1) = LENGTH OF USERS ARRAY WORK                        4Ø9Ø
C              IWORK(2) = LENGTH OF USERS ARRAY IWORK                       41ØØ
C              THE TOTAL LENGTH OF IWORK MUST BE AT LEAST                   411Ø
C              NCPTS*(NPDE + 1).                                            412Ø
C OUTPUT                                                                    413Ø
C                                                                           414Ø
C THE SOLUTION VALUES ARE NOT RETURNED DIRECTLY TO THE USER BY PDECOL.      415Ø
C THE METHODS USED IN PDECOL COMPUTE BASIS FUNCTION COEFFICIENTS, SO        416Ø
C THE USER (AFTER A RETURN FROM PDECOL) MUST CALL THE PACKAGE ROUTINE       417Ø
C VALUES TO OBTAIN HIS APPROXIMATE SOLUTION VALUES AT ANY DESIRED SPACE     418Ø
C POINTS X AT THE TIME T = TOUT.  SEE THE COMMENTS IN SUBROUTINE VALUES     419Ø
C FOR DETAILS ON HOW TO PROPERLY MAKE THE CALL.                             42ØØ
C                                                                           421Ø
C EXECUTION ERROR MESSAGES WILL BE PRINTED BY PDECOL ON LOGICAL UNIT        422Ø
C LOUT WHICH IS THE ONLY VARIABLE IN THE COMMON BLOCK /IOUNIT/.  A          423Ø
C DEFAULT OF LOUT = 3 IS SET IN THE BLOCK DATA.                             424Ø
C                                                                           425Ø
C THE COMMON BLOCK /GEARØ/ CONTAINS THE VARIABLES DTUSED, NQUSED,           426Ø
C NSTEP, NFE, AND NJE AND CAN BE ACCESSED EXTERNALLY BY THE USER IF         427Ø
C DESIRED.  RESPECTIVELY, IT CONTAINS THE STEP SIZE LAST USED (SUCCESS-     428Ø
C FULLY), THE ORDER LAST USED (SUCCESSFULLY), THE NUMBER OF STEPS TAKEN     429Ø
C SO FAR, THE NUMBER OF RESIDUAL EVALUATIONS (RES CALLS) SO FAR,            43ØØ
C AND THE NUMBER OF MATRIX EVALUATIONS (PSETIB CALLS) SO FAR.               431Ø
C DIFFUN CALLS ARE COUNTED IN WITH RESIDUAL EVALUATIONS.                    432Ø
C                                                                           433Ø
C THE OUTPUT PARAMETERS ARE..                                               434Ø
C   DT    =  THE STEP SIZE USED LAST, WHETHER SUCCESSFULLY OR NOT.          435Ø
C   TOUT  =  THE OUTPUT VALUE OF T.  IF INTEGRATION WAS SUCCESSFUL,         436Ø
```

```
C              AND THE INPUT VALUE OF INDEX WAS NOT 3, TOUT IS          437Ø
C              UNCHANGED FROM ITS INPUT VALUE.  OTHERWISE, TOUT         438Ø
C              IS THE CURRENT VALUE OF T TO WHICH THE INTEGRATION       439Ø
C              HAS BEEN COMPLETED.                                      44ØØ
C    INDEX =  INTEGER USED ON OUTPUT TO INDICATE RESULTS,               441Ø
C              WITH THE FOLLOWING VALUES AND MEANINGS..                 442Ø
C         Ø    INTEGRATION WAS COMPLETED TO TOUT OR BEYOND.            443Ø
C        -1    THE INTEGRATION WAS HALTED AFTER FAILING TO PASS THE    444Ø
C              ERROR TEST EVEN AFTER REDUCING DT BY A FACTOR OF         445Ø
C              1.E1Ø FROM ITS INITIAL VALUE.                            446Ø
C        -2    AFTER SOME INITIAL SUCCESS, THE INTEGRATION WAS          447Ø
C              HALTED EITHER BY REPEATED ERROR TEST FAILURES OR BY      448Ø
C              A TEST ON EPS.  TOO MUCH ACCURACY HAS BEEN REQUESTED.    449Ø
C        -3    THE INTEGRATION WAS HALTED AFTER FAILING TO ACHIEVE      45ØØ
C              CORRECTOR CONVERGENCE EVEN AFTER REDUCING DT BY A        451Ø
C              FACTOR OF 1.E1Ø FROM ITS INITIAL VALUE.                  452Ø
C        -4    SINGULAR MATRIX ENCOUNTERED.  PROBABLY DUE TO STORAGE    453Ø
C              OVERWRITES.                                              454Ø
C        -5    INDEX WAS 4 ON INPUT, BUT THE DESIRED CHANGES OF         455Ø
C              PARAMETERS WERE NOT IMPLEMENTED BECAUSE TOUT             456Ø
C              WAS NOT BEYOND T.  INTERPOLATION TO T = TOUT WAS         457Ø
C              PERFORMED AS ON A NORMAL RETURN.  TO TRY AGAIN,          458Ø
C              SIMPLY CALL AGAIN WITH INDEX = 4 AND A NEW TOUT.         459Ø
C        -6    ILLEGAL INDEX VALUE.                                     46ØØ
C        -7    ILLEGAL EPS VALUE.                                       461Ø
C        -8    AN ATTEMPT TO INTEGRATE IN THE WRONG DIRECTION.  THE     462Ø
C              SIGN OF DT IS WRONG RELATIVE TO TØ AND TOUT.             463Ø
C        -9    DT .EQ. Ø.Ø.                                             464Ø
C       -1Ø    ILLEGAL NINT VALUE.                                      465Ø
C       -11    ILLEGAL KORD VALUE.                                      466Ø
C       -12    ILLEGAL NCC VALUE.                                       467Ø
C       -13    ILLEGAL NPDE VALUE.                                      468Ø
C       -14    ILLEGAL MF VALUE.                                        469Ø
C       -15    ILLEGAL BREAKPOINTS - NOT STRICTLY INCREASING.           47ØØ
C       -16    INSUFFICIENT STORAGE FOR WORK OR IWORK.                  471Ø


      SUBROUTINE VALUES(X,USOL,SCTCH,NDIM1,NDIM2,NPTS,NDERV,WORK)       925Ø
C-----------------------------------------------------------------------  926Ø
C SUBROUTINE VALUES COMPUTES THE SOLUTION U AND THE FIRST NDERV          927Ø
C DERIVATIVES OF U AT THE NPTS POINTS X AND AT TIME TOUT AND RETURNS     928Ø
C THEM IN THE ARRAY USOL.  THIS ROUTINE MUST BE USED TO OBTAIN           929Ø
C SOLUTION VALUES SINCE PDECOL DOES NOT RETURN ANY SOLUTION VALUES       93ØØ
C TO THE USER.  SEE PDECOL.                                              931Ø
C                                                                        932Ø
C THE CALLING PARAMETERS ARE...                                          933Ø
C   X     =  AN ARBITRARY VECTOR OF SPATIAL POINTS OF LENGTH NPTS AT     934Ø
C            WHICH THE SOLUTION AND THE FIRST NDERV DERIVATIVE VALUES    935Ø
C            ARE TO BE CALCULATED.  IF X .LT. XLEFT ( X .GT. XRIGHT )    936Ø
C            THEN THE PIECEWISE POLYNOMIAL OVER THE LEFTMOST ( RIGHT-    937Ø
C            MOST ) INTERVAL IS EVALUATED TO CALCULATE THE SOLUTION      938Ø
C            VALUES AT THIS UNUSUAL VALUE OF  X.  SEE PDECOL.            939Ø
C                                                                        94ØØ
C   USOL  =  AN ARRAY WHICH CONTAINS THE SOLUTION AND THE FIRST          941Ø
C            NDERV DERIVATIVES OF THE SOLUTION AT ALL THE POINTS IN      942Ø
C            THE INPUT VECTOR X.  IN PARTICULAR, USOL(J,I,K) CONTAINS    943Ø
C            THE VALUE OF THE (K-1)-ST DERIVATIVE OF THE J-TH PDE        944Ø
C            COMPONENT AT THE I-TH POINT OF THE X VECTOR FOR             945Ø
C            J = 1 TO NPDE, I = 1 TO NPTS, AND K = 1 TO NDERV+1.         946Ø
C                                                                        947Ø
C   SCTCH =  A USER SUPPLIED WORKING STORAGE ARRAY OF LENGTH AT LEAST    948Ø
C            KORD*(NDERV+1).  SEE BELOW AND PDECOL FOR DEFINITIONS OF    949Ø
C            THESE PARAMETERS.                                           95ØØ
C                                                                        951Ø
C   NDIM1 =  THE FIRST DIMENSION OF THE OUTPUT ARRAY USOL IN THE CALLING 952Ø
C            PROGRAM.  NDIM1 MUST BE .GE. NPDE.                          953Ø
C                                                                        954Ø
C   NDIM2 =  THE SECOND DIMENSION OF THE OUTPUT ARRAY USOL IN THE        955Ø
C            CALLING PROGRAM.  NDIM2 MUST BE .GE. NPTS.                  956Ø
C                                                                        957Ø
C   NPTS  =  THE NUMBER OF POINTS IN THE X VECTOR.                       958Ø
C                                                                        959Ø
C   NDERV =  THE NUMBER OF DERIVATIVE VALUES OF THE SOLUTION THAT ARE    96ØØ
C            TO BE CALCULATED.  NDERV SHOULD BE LESS THAN KORD SINCE     961Ø
```

```
C            THE KORD-TH DERIVATIVE OF A POLYNOMIAL OF DEGREE KORD-1        962Ø
C            IS EQUAL TO ZERO.  SEE PDECOL.                                 963Ø
C                                                                           964Ø
C   WORK  =  THE USERS WORKING STORAGE ARRAY WHICH IS USED IN THIS CASE     965Ø
C            TO PROVIDE THE CURRENT BASIS FUNCTION COEFFICIENTS AND THE     966Ø
C            PIECEWISE POLYNOMIAL BREAKPOINT SEQUENCE.                      967Ø
C                                                                           968Ø
C PACKAGE ROUTINES CALLED..  BSPLVD,INTERV                                  969Ø
C USER ROUTINES CALLED..     NONE                                           97ØØ
C CALLED BY..                USERS MAIN PROGRAM                             971Ø
C FORTRAN FUNCTIONS USED..   NONE                                           972Ø
C                                                                           973Ø
C-------------------------------------------------------------------------  974Ø


      BLOCK DATA                                                            995Ø
C-------------------------------------------------------------------------  996Ø
C IN THE FOLLOWING DATA STATEMENT, SET..                                    997Ø
C    LOUT   = THE LOGICAL UNIT NUMBER FOR THE OUTPUT OF MESSAGES DURING     998Ø
C             THE INTEGRATION.                                              999Ø
C    NOGAUS = SET .EQ. 1 IF THE GAUSS-LEGENDRE COLLOCATION POINTS ARE      1ØØØØ
C             NOT DESIRED WHEN NCC = 2 (SEE PDECOL AND COLPNT).            1ØØ1Ø
C    MAXDER = SET .EQ. 5.  ITS VALUE REPRESENTS THE MAXIMUM ORDER OF       1ØØ2Ø
C             THE TIME INTEGRATION ALLOWED.  ITS VALUE AFFECTS THE STOR-   1ØØ3Ø
C             AGE REQUIRED IN   WORK   AND MAY BE CHANGED IF DESIRED       1ØØ4Ø
C             (SEE COSET FOR RESTRICTIONS).                                1ØØ5Ø
C    UROUND = THE UNIT ROUNDOFF OF THE MACHINE, I.E. THE SMALLEST          1ØØ6Ø
C             POSITIVE U SUCH THAT 1. + U .NE. 1. ON THE MACHINE.          1ØØ7Ø
C-------------------------------------------------------------------------  1ØØ8Ø
      COMMON /GEAR1/ DUM(5),UROUND,IDUM(4)                                  1ØØ9Ø
      COMMON /OPTION/ NOGAUS,MAXDER                                         1Ø1ØØ
      COMMON /IOUNIT/ LOUT                                                  1Ø11Ø
      DATA LOUT,NOGAUS,MAXDER,UROUND/3,Ø,5,7.1E-15/                         1Ø12Ø
      END                                                                   1Ø13Ø
```

ALGORITHM 541
Efficient Fortran Subprograms for the Solution of Separable Elliptic Partial Differential Equations [D3]

PAUL N. SWARZTRAUBER
National Center for Atmospheric Research
and
ROLAND A. SWEET
University of Colorado at Denver

Key Words and Phrases: elliptic partial differential equations, software, linear systems
CR Categories: 4.19, 5.14, 5.17
Language: Fortran

DESCRIPTION

1. Introduction

Computer models of geophysical processes often require the numerical solution of elliptic partial differential equations. This is particularly true for models that make use of stream functions, velocity potentials, or vorticity equations and for models that compute the pressure of an incompressible fluid. The numerical solution of elliptic equations can be a formidable programming task. Moreover, the equations are often time dependent, requiring repeated solutions and, hence, considerable computing resources.

Recent advances in computing methods [1, 2] made it possible to solve a very large class of elliptic equations (the separable ones) rapidly and with minimal storage. And as a result of work on singular problems [6, 8], this class is free of special cases for which solutions cannot be obtained numerically. This paper describes a package of computer programs that make use of current methods for solving elliptic partial differential equations. The package is fully documented in [7].

We first became involved in implementing the Buneman algorithm [2] and its extensions via the capacitance matrix approach [1] for solving Poisson's equation in polar coordinates on a disk. This led directly to two important problems: the need to treat the coordinate singularity which occurs at the center of the disk [8] and the restriction that the number of unknowns in one coordinate must be a power of 2 [10]. It is desirable to relax that restriction for certain studies, since geophysicists often work on grids that result in a number of points which are not

Received 10 February 1977.

Authors' addresses: P.N. Swarztrauber, National Center for Atmospheric Research, P.O. Box 3000, Boulder, CO 80307; R.A. Sweet, University of Colorado at Denver, Denver, CO 80202.

a power of 2; for example, the commonly used 5° grid results in 72 grid points in longitude.

This work demonstrated that a Poisson equation in any of the usual coordinate systems was readily solvable, except on the interior of the sphere. The equation for that problem is a special case of the general separable equation (see Section 2, subroutine S.7) and cannot be solved by the Buneman algorithm. This difficulty led to the development of a cyclic reduction algorithm for separable elliptic equations [5]. We then wrote two subroutines, POIS and BLKTRI, to implement the cyclic reduction and generalized cyclic reduction, respectively. But because users were significantly inconvenienced by having to set up the finite-difference approximations before using either of these subroutines, the routines were generally *not* well accepted. Hence the National Center for Atmospheric Research (NCAR) supported the development of the driver programs S.1 through S.5 (listed in Secton 2), which automated the process of generating the finite-difference equations for Poisson's equation in various coordinate systems. After these seven subroutines had been developed, we wrote a technical note [7] describing their use in detail. The combination of the ease of use of the subroutines and the complete documentation substantially increased the usage of the subroutines.

This experience made it quite apparent that even if a numerical method is excellent, the programs will not be widely used unless the software is easy to use and the documentation is quite clear. In addition to supporting user convenience, our philosophy has been to avoid terminology that might limit the class of users. For example, terms such as Dirichlet and Neumann boundary conditions, which tend to limit readership, have been replaced with either the solution or the derivative of the solution being specified.

The NCAR technical note [7] contains seven chapters, each describing one of the seven Fortran subroutines, and an Appendix dealing with least-squares solution of singular linear systems of equations. The first five subroutines solve a Helmholtz equation in Cartesian, polar, cylindrical, interior spherical, and surface spherical coordinates, respectively. The equations are solved on regular domains, namely, on a rectangle in the particular coordinate system that is chosen. For example, in polar coordinates the user can select the domain $a \leq r \leq b$ and $c \leq \theta \leq d$, where a can be zero. The user can also choose any of the standard boundary conditions. The solution or its derivative can be specified on the boundary, or periodic boundary conditions can be specified. Although these programs solve only two-dimensional problems, they can be adapted for use in three-dimensional problems by Fourier transforming in the third variable and using the parameter in the Helmholtz equation. This procedure is described in detail in the technical note [7]. Each chapter also contains a sample Fortran program which illustrates the use of the subroutine and provides a check during program installation.

In addition to the description of the Fortran subroutines, the first five chapters contain the finite-difference approximations used for the Helmholtz equation and the boundary conditions. Any special equations at singular points of the Helmholtz equation (e.g. at the origin, $r = 0$) are also presented. Approximations of second-order accuracy are used throughout. Each of these five chapters contains a section entitled "Singular Problems," which augments the Appendix in describing how solutions are obtained in the least-squares sense when a solution does not exist in the usual sense.

Each of these five subroutines calls one of the last two subroutines (POIS and BLKTRI), which can be used to solve a more general class of equation. Chapter VI describes subroutine POIS, which can be used to solve finite-difference approximations to an equation of the form

$$a(x)\frac{\partial^2 u}{\partial x^2} + b(x)\frac{\partial u}{\partial x} + c(x)u + \frac{\partial^2 u}{\partial y^2} = g(x, y).$$

Chapter VII describes subroutine BLKTRI, which can be used to solve finite-difference approximations to the general separable equation

$$a(x)\frac{\partial^2 u}{\partial x^2} + b(x)\frac{\partial u}{\partial x} + c(x)u + d(y)\frac{\partial^2 u}{\partial y^2} + e(y)\frac{\partial u}{\partial y} + f(y)u = g(x, y).$$

Although BLKTRI can solve problems that POIS solves, POIS is faster.

Subroutines POIS and BLKTRI use cyclic reduction and generalized cyclic reduction, respectively [2, 5, 10], to solve the large sparse linear systems. Although some discussion of these methods is included, program descriptions are presented separately for those who may wish to use the package only as a computational aid.

In Section 2 of this paper we briefly describe the capabilities of each of the seven subroutines; in Section 3 we summarize the various sections in the documentation; and in Section 4 we illustrate the documentation by presenting a small portion of the NCAR technical note [7]. And finally, in Section 5 we describe the modifications that have been made since the initial publication of the package and several developments that we are now planning or implementing.

2. Software Package

The package consists of seven subroutiness The first five, which are referred to as drivers, generate finite-difference approximations to a two-dimensional Helmholtz equation in a particular coordinate system. They are the following.

S.1 Subroutine PWSCRT solves the Helmholtz equation in Cartesian coordinates

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \lambda u = f(x, y)$$

on the rectangle $a \le x \le b,\ c \le y \le d$.

S.2 Subroutine PWSPLR solves the Helmholtz equation in polar coordinates

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} + \lambda u = f(r, \theta)$$

on the disk (or sector of the disk) $0 \le a \le r \le b,\ 0 \le c \le \theta \le d \le 2\pi$.

S.3 Subroutine PWSCYL solves the modified Helmholtz equation in cylindrical coordinates

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) + \frac{\partial^2 u}{\partial z^2} + \frac{\lambda}{r^2}u = f(r, z)$$

on the slab of the cylinder $0 \le a \le r \le b,\ c \le z \le d$.

S.4 Subroutine PWSSSP solves the Helmholtz equation in spherical surface coordinates

$$\frac{1}{\sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta\frac{\partial u}{\partial \theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2 u}{\partial \phi^2} + \lambda u = f(\theta, \phi)$$

on the sphere (or sector of the sphere) $0 \le a \le \theta \le b \le \pi,\ 0 \le c \le \phi \le d \le 2\pi$.

S.5 Subroutine PWSCSP solves the modified Helmholtz equation in interior spherical coordinates

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial u}{\partial r}\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial \theta}\left(\sin\theta\frac{\partial u}{\partial \theta}\right) + \frac{\lambda}{r^2\sin^2\theta}u = f(\theta, r)$$

on all or a portion of a cross section of the sphere $0 \le a \le r \le b,\ 0 \le c \le \theta \le d \le \pi$.

On each of the boundaries, any of the following conditions can be specified.

B.1 The solution is specified (Dirichlet condition).

B.2 The derivative of the solution is specified (Neumann condition).

B.3 The solution is specified to be periodic.

B.4 The solution is unspecified (this condition applies when the domain of the equation includes a singularity of the coordinate system, e.g. when solving the Helmholtz equation in polar coordinates on the entire disk $0 \le r \le b$, $0 \le \theta \le 2\pi$).

Each of the five subroutines performs the following tasks.

T.1 defines the coefficients in the linear system of equations resulting from the finite-difference approximation.

T.2 incorporates the given boundary data into the right side of the equations.

T.3 perturbs the right side of the equation when the linear system is singular in order to guarantee that a solution exists.

Once the linear system has been defined, each of the routines calls one of the two following subroutines for solving linear systems.

S.6 Subroutine POIS solves systems resulting from finite-difference approximations to equations of the form

$$a(x)\frac{\partial^2 u}{\partial x^2} + b(x)\frac{\partial u}{\partial x} + c(x)u + \frac{\partial^2 u}{\partial y^2} = f(x, y).$$

S.7 Subroutine BLKTRI solves systems resulting from finite-difference approximations to general separable elliptic equations of the form

$$a(x)\frac{\partial^2 u}{\partial x^2} + b(x)\frac{\partial u}{\partial x} + c(x)u + d(y)\frac{\partial^2 u}{\partial y^2} + e(y)\frac{\partial u}{\partial y} + f(y)u = g(x, y).$$

3. Documentation of the Package

The documentation is completely contained in the NCAR technical note [7]. Each of the seven chapters is devoted to one subroutine and contains the following sections.

D.1 Calling Sequence and Parameter List Description. This section gives detailed descriptions of the parameters on input and output and information likely to be of value to a user deciding whether or not he can use the routine (e.g. memory space required, timing, and accuracy). This section will be of primary interest to those who wish to use the program strictly as a computational aid. This information is also contained on COMMENT cards at the beginning of each subroutine.

D.2 Difference Approximations. This section gives the detailed development of the difference approximations used for the equations, for the boundary conditions, and for the equations at coordinate singularities.

D.3 Singular Problems. This section gives the exact method used to perturb the right side of the equation when the user has specified a problem resulting in a singular linear system of equations.

D.4 Three-Dimensional Problems. This section demonstrates how to combine the use of the Fourier method with the routine in order to solve a Poisson equation in three space variables. (It is for this reason that the Helmholtz term involving λ sometimes has a function multiplying it.)

D.5 Sample Problem. This section gives a complete description of how to define input parameters for a selected differential equation. A Fortran implementation of the description is also given and certain output variables are given. These programs are included in the package as an aid to the user when the package is installed at his institution.

The documentation clearly states the locations of all machine-dependent constants used, e.g. π, and all required library routines, e.g. COS(X). The machine precision is required by BLKTRI, but it is computed (approximately) internally.

4. Sample Documentation

In this section we reproduce selected parts of the documentation for the routine PWSSSP to illustrate the use of the routine and the format of the documentation. We shall first state a continuous problem to be solved and then give a Fortran program that defines all the necessary input parameters. We then present the

formal parameter list description. Most of the following material is taken verbatim from Chapter V of the NCAR technical note [7], although in a different order.

Assume that we wish to approximate the solution of the Poisson equation

$$\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial u}{\partial\theta}\right)+\frac{1}{\sin^2\theta}\frac{\partial^2 u}{\partial\phi^2}=2-6\sin^2\theta\sin^2\phi$$

on the northern hemisphere $0 \le \theta \le \pi/2$, $0 \le \phi \le 2\pi$, subject to equatorial symmetry, i.e. $(\partial u/\partial\theta)(\pi/2, \phi) = 0$, $0 \le \phi \le 2\pi$.

The exact solution to this problem is

$$u(\theta, \phi) = \sin^2\theta\sin^2\phi + c,$$

where c is any constant. Hence, this is a singular problem.

We choose a 5° grid for our approximation. The Fortran program that defines the necessary input parameters for subroutine PWSSSP is given in the first program of the listings which follow.

```
C                                                                       POIØ5Ø42
C     PROGRAM TO ILLUSTRATE THE USE OF PWSSSP                           POIØ5Ø43
C                                                                       POIØ5Ø44
      DIMENSION       F(19,73)    ,BDTF(73)    ,SINT(19)    ,SINP(73)    ,  POIØ5Ø45
     1                W(647)                                             POIØ5Ø46
C                                                                       POIØ5Ø47
C     THE VALUE OF IDIMF IS THE FIRST DIMENSION OF F. W IS DIMENSIONED   POIØ5Ø48
C     11*(M+1)+6X(N+1)=647 SINCE M=18 AND N=72                          POIØ5Ø49
C                                                                       POIØ5Ø5Ø
      PI = 3.141592653589793                                            POIØ5Ø51
      INTL = Ø                                                          POIØ5Ø52
      TS = Ø                                                            POIØ5Ø53
      TF = PI/2.                                                        POIØ5Ø54
      M = 18                                                            POIØ5Ø55
      MBDCND = 6                                                        POIØ5Ø56
      PS = Ø                                                            POIØ5Ø57
      PF = PI+PI                                                        POIØ5Ø58
      N = 72                                                            POIØ5Ø59
      NBDCND = Ø                                                        POIØ5Ø6Ø
      ELMBDA = Ø.                                                       POIØ5Ø61
      IDIMF = 19                                                        POIØ5Ø62
C                                                                       POIØ5Ø63
C     GENERATE SINES FOR USE IN SUBSEQUENT COMPUTATIONS                 POIØ5Ø64
C                                                                       POIØ5Ø65
      DTHETA = TF/FLOAT(M)                                              POIØ5Ø66
      MP1 = M+1                                                         POIØ5Ø67
      DO 1Ø1 I=1,MP1                                                    POIØ5Ø68
         SINT(I) = SIN(FLOAT(I-1)*DTHETA)                               POIØ5Ø69
  1Ø1 CONTINUE                                                          POIØ5Ø7Ø
      DPHI = (PI+PI)/FLOAT(N)                                           POIØ5Ø71
      NP1 = N+1                                                         POIØ5Ø72
      DO 1Ø2 J=1,NP1                                                    POIØ5Ø73
         SINP(J) = SIN(FLOAT(J-1)*DPHI)                                 POIØ5Ø74
  1Ø2 CONTINUE                                                          POIØ5Ø75
C                                                                       POIØ5Ø76
C     COMPUTE RIGHT SIDE OF EQUATION AND STORE IN F                     POIØ5Ø77
C                                                                       POIØ5Ø78
      DO 1Ø4 J=1,NP1                                                    POIØ5Ø79
         DO 1Ø3 I=1,MP1                                                 POIØ5Ø8Ø
            F(I,J) = 2.-6.*(SINT(I)*SINP(J))**2                         POIØ5Ø81
  1Ø3    CONTINUE                                                       POIØ5Ø82
  1Ø4 CONTINUE                                                          POIØ5Ø83
C                                                                       POIØ5Ø84
C     STORE DERIVATIVE DATA AT THE EQUATOR                              POIØ5Ø85
C                                                                       POIØ5Ø86
      DO 1Ø5 J=1,NP1                                                    POIØ5Ø87
         BDTF(J) = Ø.                                                   POIØ5Ø88
  1Ø5 CONTINUE                                                          POIØ5Ø89
C                                                                       POIØ5Ø9Ø
      CALL PWSSSP (INTL,TS,TF,M,MBDCND,BDTS,BDTF,PS,PF,N,NBDCND,BDPS,   POIØ5Ø91
     1             BDPF,ELMBDA,F,IDIMF,PERTRB,IERROR,W)                 POIØ5Ø92
C                                                                       POIØ5Ø93
C     COMPUTE DISCRETIZATION ERROR. SINCE PROBLEM IS SINGULAR, THE      POIØ5Ø94
C     SOLUTION MUST BE NORMALIZED.                                      POIØ5Ø95
```

```
C                                                                       POIØ5Ø96
      ERR = Ø                                                           POIØ5Ø97
      DO 1Ø7 J=1,NP1                                                    POIØ5Ø98
         DO 1Ø6 I=1,MP1                                                 POIØ5Ø99
            Z = ABS(F(I,J)-(SINT(I)*SINP(J))**2-F(1,1))                 POIØ51ØØ
            IF (Z .GT. ERR) ERR = Z                                     POIØ51Ø1
  1Ø6    CONTINUE                                                       POIØ51Ø2
  1Ø7 CONTINUE                                                          POIØ51Ø3
C                                                                       POIØ51Ø4
      PRINT 1ØØ1 , IERROR,ERR,PERTRB                                    POIØ51Ø5
      STOP                                                              POIØ51Ø6
C                                                                       POIØ51Ø7
C                                                                       POIØ51Ø8
C                                                                       POIØ51Ø9
 1ØØ1 FORMAT (///9H IERROR= ,I3,1ØX,22H DISCRETIZATION ERR = ,E12.4,    POIØ511Ø
     1        14H PERTURBATION=,E12.4)                                  POIØ5111
C                                                                       POIØ5112
      END                                                               POIØ5113
```

After the CALL to PWSSSP, the discretization error—the maximum absolute difference between the exact solution and the finite-difference approximation—is computed and printed along with the output parameter PERTRB. The printed output is

IERROR = 0

DISCRETIZATION ERR = 3.3811E − 03 PERTURBATION = 6.3220E − 04

The parameters in the call to PWSSSP are defined in the listing which appears at the end of this paper.

Accuracy and Timing. The execution time is proportional to MN $\log_2$ N and is given in Table I for the NCAR Control Data 7600 computer. To test the accuracy of the method, a uniform random number generator was used to create an array V(I, J), where $0 \leq V(I, J) \leq 1$. An array F(I, J) was then computed by differencing V(I, J) in double precision, using the difference equations that correspond to the sample problem. With F(I, J) as a right side, subroutine PWSSSP was used to compute a solution U(I, J). The maximum absolute difference between U(I, J) and V(I, J) is given in Table I. These errors can be compared with the accuracy of the NCAR 7600 (10^{-14}) to give a measure of the loss of significant digits due to roundoff. This error should not be confused with the discretization error given after the sample Fortran program.

5. Concluding Remarks

This package is not static, but is currently undergoing modifications to improve the solution techniques and extend the range of problems that can be solved. The following additions and modifications have already been made.

(1) Currently available is a subroutine, SEPELI, which discretizes the general separable elliptic equation. Thus the process of differencing the equations, incorporating the boundary conditions, and solving the resulting equations is automated as in the driver programs described in Section 2. Also included in this program is the option of obtaining fourth-order accurate solutions via the method of deferred corrections [3]. As in the other drivers, a weighted least-squares solution is obtained if that solution does not exist in the usual sense. More

Table I

M	N	Execution time (ms)	Maximum absolute difference
8	32	14	2.3×10^{-13}
16	64	60	1.3×10^{-12}
32	128	248	7.7×10^{-12}

Note: If NBDCND $\neq 0$, the execution times can be halved.

information may be obtained from John Adams, who has developed this program at NCAR.

(2) Replacements for the subroutines POIS and BLKTRI have been developed and tested and are in their final stage of documentation. The earlier versions of POIS and BLKTRI impose restrictions on the block order of the linear system of equations. The new algorithms eliminate this restriction. They also reduce to the efficient cyclic reduction algorithms when $N = 2^p$. The new algorithm for POIS is given in [9].

(3) With the use of this new algorithm, a subroutine, POISTG, has been developed [4] to solve finite-difference approximations to Helmholtz equations defined on staggered grids. This routine has been written in the format of POIS and BLKTRI. Drivers such as those described in Section 2 are now being written to solve finite-difference approximations to modified Helmholtz equations defined on a staggered grid.

In the early planning stage is a subroutine that will solve the general nonseparable linear elliptic equation in which the coefficients depend on both independent variables. However, it is anticipated that this program will use iterative rather than direct methods.

We feel that this package of high-quality, reliable, efficient programs for the solution of finite-difference approximations to elliptic equations is a significant step in the development of a complete package designed to solve an extremely wide class of elliptic partial differential equations.

NOTE ADDED IN PROOF. In the period since this paper was submitted for publication, all of the subroutines listed in modifications (1)–(3) above have been implemented in the latest version of the package. In addition, a driver for solving a three-dimensional Helmholtz equation in Cartesian coordinates, its associated solver, and two fast Fourier transform packages have been implemented. The interested reader should contact one of the authors at NCAR.

REFERENCES

1. BUZBEE, B.L., DORR, F.W., GEORGE, J.A., AND GOLUB, G. H. The direct solution of the discrete Poisson equation on irregular regions. *SIAM J. Numer. Anal. 8* (1971), 722–736.
2. BUZBEE, B.L., GOLUB, G.H., AND NIELSON, C.W. On direct methods for solving Poisson's equations. *SIAM J. Numer. Anal. 7* (1970), 627–656.
3. PEREYERA, V. Higher order difference solution of differential equations. Rep. STAN-CS-73-348, Comptr. Sci. Dept., Stanford U., Stanford, Calif.
4. SCHUMANN, U., AND SWEET, R.A. A direct mmthod for the solution of Poisson's equation with Neumann boundary conditions on a staggered grid of arbitrary size. *J. Comput. Phys. 20* (1976), 171–182.
5. SWARZTRAUBER, P.N. A direct method for the discrete solution of separable elliptic equations. *SIAM J. Numer. Anal. 11* (1974), 1136–1150.
6. SWARZTRAUBER, P.N. The direct solution of the discrete Poisson equation on the surface of a sphere. *J. Comput. Phys. 15* (1974), 46–54.
7. SWARZTRAUBER, P.N., AND SWEET, R. A. Efficient FORTRAN subprograms for the solution of elliptic partial differential equations. Tech. Note TN/IA-109, Nat. Ctr. for Atmospheric Res., Boulder, Colo., 1975, and Errata, 1976. Also available from the Nat. Tech. Inform. Service as document PB263 498/AS in paper ($6.00) or on microfiche ($3.00).
8. SWARZTRAUBER, P.N., AND SWEET, R.A. The direct solution of the discrete Poisson equation on a disk. *SIAM J. Numer. Anal. 10* (1973), 900–907.
9. SWEET, R.A. A cyclic reduction algorithm for block tridiagonal systems of arbitrary dimension. *SIAM J. Numer. Anal. 14* (1977), 706–720.
10. SWEET, R.A. A generalized cyclic reduction algorithm. *SIAM J. Numer. Anal. 11* (1974), 506–520.

ALGORITHM

[Summary information and a part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service.]

NAME(n): indicates a Fortran module with n records
NAME^T(n): indicates "NAME" is included for testing purposes
Contents: PWSCRT(416), PWSPLR(512), PWSCYL(455), PWSSSP(313), PWSSS1(356), PWSCSP(340), PWSCS1(247), POIS(153), POISGN(411), POINIT(136), TRID(43), TPIDP(63), BLKTRI(200), BLKTRI(132), PROD(90), PRODP(107), CPROD(103), CPRODP(125), PPADD(119), PSGF(12), BSRH(17), PPSGF(9), PPSPF(9), COMPB(98), TQLRAT(162), NCHECK(29), APEXPS(9), STORE(5), MAIN1^T(101), MAIN2^T(101), MAIN3^T(114), MAIN4^T(95), MAIN5^T(140), MAIN6^T(110), MAIN7^T(117)

```
      SUBROUTINE PWSSSP (INTL,TS,TF,M,MBDCND,BDTS,BDTF,PS,PF,N,NBDCND,  POIØ1387
     1                   BDPS,BDPF,ELMBDA,F,IDIMF,PERTRB,IERROR,W)      POIØ1388
C                                                                       POIØ1389
C     SUBROUTINE PWSSSP SOLVES A FINITE DIFFERENCE APPROXIMATION TO THE POIØ1414
C     HELMHOLTZ EQUATION IN SPHERICAL COORDINATES AND ON THE SURFACE OF POIØ1415
C     THE UNIT SPHERE (RADIUS OF 1):                                    POIØ1416
C                                                                       POIØ1417
C          (1/SIN(THETA))(D/DTHETA)(SIN(THETA)(D/DTHETA)U)              POIØ1418
C                                                                       POIØ1419
C             + (1/SIN(THETA)**2)(D/DPHI)(D/DPHI)U                      POIØ142Ø
C                                                                       POIØ1421
C             + LAMBDA*U = F(THETA,PHI)                                 POIØ1422
C                                                                       POIØ1423
C     WHERE THETA IS COLATITUDE AND PHI IS LONGITUDE.                   POIØ1424
C                                                                       POIØ1425
C     THE ARGUMENTS ARE DEFINED AS:                                     POIØ1426
C                                                                       POIØ1427
C                                                                       POIØ1428
C     * * * * * * * * * *     ON INPUT     * * * * * * * * * *          POIØ1429
C                                                                       POIØ143Ø
C     INTL                                                              POIØ1431
C       = Ø  ON INITIAL ENTRY TO PWSSSP OR IF N,NBDCND,PS OR PF ARE     POIØ1432
C            CHANGED  FROM A PREVIOUS CALL                              POIØ1433
C       = 1  IF PS,PF,N AND NBDCND ARE UNCHANGED FROM A PREVIOUS CALL   POIØ1434
C                                                                       POIØ1435
C       NOTE:  A CALL WITH INTL = 1 IS ABOUT 1 PERCENT FASTER THAN A    POIØ1436
C              CALL WITH INTL = Ø  .                                    POIØ1437
C                                                                       POIØ1438
C     TS,TF                                                             POIØ1439
C       THE RANGE OF THETA (COLATITUDE), I.E., TS .LE. THETA .LE. TF.   POIØ144Ø
C       TS MUST BE LESS THAN TF.  TS AND TF ARE IN RADIANS.  A TS OF    POIØ1441
C       ZERO CORRESPONDS TO THE NORTH POLE AND A TF OF PI CORRESPONDS TOPOIØ1442
C       THE SOUTH POLE.                                                 POIØ1443
C                                                                       POIØ1444
C     M                                                                 POIØ1445
C       THE NUMBER OF PANELS INTO WHICH THE INTERVAL (TS,TF) IS         POIØ1446
C       SUBDIVIDED.  HENCE, THERE WILL BE M+1 GRID POINTS IN THE        POIØ1447
C       THETA-DIRECTION GIVEN BY THETA(I) = (I-1)DTHETA+TS FOR          POIØ1448
C       I = 1,2,...,M+1, WHERE DTHETA = (TF-TS)/M IS THE PANEL WIDTH.   POIØ1449
C                                                                       POIØ145Ø
C     MBDCND                                                            POIØ1451
C       INDICATES THE TYPE OF BOUNDARY CONDITION AT THETA = TS AND      POIØ1452
C       THETA = TF.                                                     POIØ1453
C                                                                       POIØ1454
C       = 1  IF THE SOLUTION IS SPECIFIED AT THETA = TS AND THETA = TF. POIØ1455
C       = 2  IF THE SOLUTION IS SPECIFIED AT THETA = TS AND THE         POIØ1456
C            DERIVATIVE OF THE SOLUTION WITH RESPECT TO THETA IS        POIØ1457
C            SPECIFIED AT THETA = TF (SEE NOTE 2 BELOW).                POIØ1458
C       = 3  IF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO THETA IS POIØ1459
C            SPECIFIED AT THETA = TS AND THETA = TF (SEE NOTES 1,2      POIØ146Ø
C            BELOW).                                                    POIØ1461
C       = 4  IF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO THETA IS POIØ1462
C            SPECIFIED AT THETA = TS (SEE NOTE 1 BELOW) AND THE         POIØ1463
C            SOLUTION IS SPECIFIED AT THETA = TF.                       POIØ1464
C       = 5  IF THE SOLUTION IS UNSPECIFIED AT THETA = TS = Ø AND THE   POIØ1465
C            SOLUTION IS SPECIFIED AT THETA = TF.                       POIØ1466
C       = 6  IF THE SOLUTION IS UNSPECIFIED AT THETA = TS = Ø AND THE   POIØ1467
C            DERIVATIVE OF THE SOLUTION WITH RESPECT TO THETA IS        POIØ1468
C            SPECIFIED AT THETA = TF (SEE NOTE 2 BELOW).                POIØ1469
C       = 7  IF THE SOLUTION IS SPECIFIED AT THETA = TS AND THE         POIØ147Ø
C            SOLUTION IS UNSPECIFIED AT THETA = TF = PI.                POIØ1471
```

```
C       = 8  IF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO THETA IS POIØ1472
C            SPECIFIED AT THETA = TS (SEE NOTE 1 BELOW) AND THE         POIØ1473
C            SOLUTION IS UNSPECIFIED AT THETA = TF = PI.                POIØ1474
C       = 9  IF THE SOLUTION IS UNSPECIFIED AT THETA = TS = Ø AND       POIØ1475
C            THETA = TF = PI.                                           POIØ1476
C                                                                       POIØ1477
C       NOTES:  1.  IF TS = Ø, DO NOT USE MBDCND = 3,4, OR 8, BUT       POIØ1478
C                   INSTEAD USE MBDCND = 5,6, OR 9  .                   POIØ1479
C               2.  IF TF = PI, DO NOT USE MBDCND = 2,3, OR 6, BUT      POIØ148Ø
C                   INSTEAD USE MBDCND = 7,8, OR 9  .                   POIØ1481
C                                                                       POIØ1482
C     BDTS                                                              POIØ1483
C       A ONE-DIMENSIONAL ARRAY OF LENGTH N+1 THAT SPECIFIES THE VALUES POIØ1484
C       OF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO THETA AT      POIØ1485
C       THETA = TS.  WHEN MBDCND = 3,4, OR 8,                           POIØ1486
C                                                                       POIØ1487
C            BDTS(J) = (D/DTHETA)U(TS,PHI(J)), J = 1,2,...,N+1  .       POIØ1488
C                                                                       POIØ1489
C       WHEN MBDCND HAS ANY OTHER VALUE, BDTS IS A DUMMY VARIABLE.      POIØ149Ø
C                                                                       POIØ1491
C     BDTF                                                              POIØ1492
C       A ONE-DIMENSIONAL ARRAY OF LENGTH N+1 THAT SPECIFIES THE VALUES POIØ1493
C       OF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO THETA AT      POIØ1494
C       THETA = TF.  WHEN MBDCND = 2,3, OR 6,                           POIØ1495
C                                                                       POIØ1496
C            BDTF(J) = (D/DTHETA)U(TF,PHI(J)), J = 1,2,...,N+1  .       POIØ1497
C                                                                       POIØ1498
C       WHEN MBDCND HAS ANY OTHER VALUE, BDTF IS A DUMMY VARIABLE.      POIØ1499
C                                                                       POIØ15ØØ
C     PS,PF                                                             POIØ15Ø1
C       THE RANGE OF PHI (LONGITUDE), I.E., PS .LE. PHI .LE. PF.  PS    POIØ15Ø2
C       MUST BE LESS THAN PF.  PS AND PF ARE IN RADIANS.  IF PS = Ø AND POIØ15Ø3
C       PF = 2*PI, PERIODIC BOUNDARY CONDITIONS ARE USUALLY PRESCRIBED. POIØ15Ø4
C                                                                       POIØ15Ø5
C     N                                                                 POIØ15Ø6
C       THE NUMBER OF PANELS INTO WHICH THE INTERVAL (PS,PF) IS         POIØ15Ø7
C       SUBDIVIDED.  HENCE, THERE WILL BE N+1 GRID POINTS IN THE        POIØ15Ø8
C       PHI-DIRECTION GIVEN BY PHI(J) = (J-1)DPHI+PS  FOR               POIØ15Ø9
C       J = 1,2,...,N+1, WHERE DPHI = (PF-PS)/N IS THE PANEL WIDTH.  N  POIØ151Ø
C       MUST BE OF THE FORM (2**P)(3**Q)(5**R) WHERE P, Q, AND R ARE ANYPOIØ1511
C       NON-NEGATIVE INTEGERS.  N MUST BE GREATER THAN 2  .             POIØ1512
C                                                                       POIØ1513
C     NBDCND                                                            POIØ1514
C       INDICATES THE TYPE OF BOUNDARY CONDITION AT PHI = PS AND        POIØ1515
C       PHI = PF.                                                       POIØ1516
C                                                                       POIØ1517
C       = Ø  IF THE SOLUTION IS PERIODIC IN PHI, I.E.,                  POIØ1518
C            U(I,1) = U(I,N+1).                                         POIØ1519
C       = 1  IF THE SOLUTION IS SPECIFIED AT PHI = PS AND PHI = PF      POIØ152Ø
C            (SEE NOTE BELOW).                                          POIØ1521
C       = 2  IF THE SOLUTION IS SPECIFIED AT PHI = PS (SEE NOTE BELOW)  POIØ1522
C            AND THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO PHI IS  POIØ1523
C            SPECIFIED AT PHI = PF.                                     POIØ1524
C       = 3  IF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO PHI IS   POIØ1525
C            SPECIFIED AT PHI = PS AND PHI = PF.                        POIØ1526
C       = 4  IF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO PHI IS   POIØ1527
C            SPECIFIED AT PS AND THE SOLUTION IS SPECIFIED AT PHI = PF  POIØ1528
C            (SEE NOTE BELOW).                                          POIØ1529
C                                                                       POIØ153Ø
C       NOTE:  NBDCND = 1,2, OR 4 CANNOT BE USED WITH                   POIØ1531
C              MBDCND = 5,6,7,8, OR 9 (THE FORMER INDICATES THAT THE    POIØ1532
C                       SOLUTION IS SPECIFIED AT A POLE, THE LATTER     POIØ1533
C                       INDICATES THAT THE SOLUTION IS UNSPECIFIED).    POIØ1534
C                       USE INSTEAD                                     POIØ1535
C              MBDCND = 1 OR 2  .                                       POIØ1536
C                                                                       POIØ1537
C     BDPS                                                              POIØ1538
C       A ONE-DIMENSIONAL ARRAY OF LENGTH M+1 THAT SPECIFIES THE VALUES POIØ1539
C       OF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO PHI AT        POIØ154Ø
C       PHI = PS.  WHEN NBDCND = 3 OR 4,                                POIØ1541
C                                                                       POIØ1542
C            BDPS(I) = (D/DPHI)U(THETA(I),PS), I = 1,2,...,M+1  .       POIØ1543
C                                                                       POIØ1544
C       WHEN NBDCND HAS ANY OTHER VALUE, BDPS IS A DUMMY VARIABLE.      POIØ1545
C                                                                       POIØ1546
C     BDPF                                                              POIØ1547
C       A ONE-DIMENSIONAL ARRAY OF LENGTH M+1 THAT SPECIFIES THE VALUES POIØ1548
```

```
C       OF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO PHI AT        POIØ1549
C       PHI = PF.  WHEN NBDCND = 2 OR 3,                                POIØ155Ø
C                                                                       POIØ1551
C            BDPF(I) = (D/DPHI)U(THETA(I),PF), I = 1,2,...,M+1  .       POIØ1552
C                                                                       POIØ1553
C       WHEN NBDCND HAS ANY OTHER VALUE, BDPF IS A DUMMY VARIABLE.      POIØ1554
C                                                                       POIØ1555
C     ELMBDA                                                            POIØ1556
C       THE CONSTANT LAMBDA IN THE HELMHOLTZ EQUATION.  IF              POIØ1557
C       LAMBDA .GT. Ø, A SOLUTION MAY NOT EXIST.  HOWEVER, PWSSSP WILL  POIØ1558
C       ATTEMPT TO FIND A SOLUTION.                                     POIØ1559
C                                                                       POIØ156Ø
C     F                                                                 POIØ1561
C       A TWO-DIMENSIONAL ARRAY THAT SPECIFIES THE VALUE OF THE RIGHT   POIØ1562
C       SIDE OF THE HELMHOLTZ EQUATION AND BOUNDARY VALUES (IF ANY).    POIØ1563
C       FOR I = 2,3,...,M  AND  J = 2,3,...,N                           POIØ1564
C                                                                       POIØ1565
C            F(I,J) = F(THETA(I),PHI(J)).                               POIØ1566
C                                                                       POIØ1567
C       ON THE BOUNDARIES F IS DEFINED BY                               POIØ1568
C                                                                       POIØ1569
C            MBDCND   F(1,J)            F(M+1,J)                        POIØ157Ø
C            ------   ------------      ------------                    POIØ1571
C                                                                       POIØ1572
C              1      U(TS,PHI(J))      U(TF,PHI(J))                    POIØ1573
C              2      U(TS,PHI(J))      F(TF,PHI(J))                    POIØ1574
C              3      F(TS,PHI(J))      F(TF,PHI(J))                    POIØ1575
C              4      F(TS,PHI(J))      U(TF,PHI(J))                    POIØ1576
C              5      F(Ø,PS)           U(TF,PHI(J))   J = 1,2,...,N+1  POIØ1577
C              6      F(Ø,PS)           F(TF,PHI(J))                    POIØ1578
C              7      U(TS,PHI(J))      F(PI,PS)                        POIØ1579
C              8      F(TS,PHI(J))      F(PI,PS)                        POIØ158Ø
C              9      F(Ø,PS)           F(PI,PS)                        POIØ1581
C                                                                       POIØ1582
C            NBDCND   F(I,1)            F(I,N+1)                        POIØ1583
C            ------   --------------    --------------                  POIØ1584
C                                                                       POIØ1585
C              Ø      F(THETA(I),PS)    F(THETA(I),PS)                  POIØ1586
C              1      U(THETA(I),PS)    U(THETA(I),PF)                  POIØ1587
C              2      U(THETA(I),PS)    F(THETA(I),PF)   I = 1,2,...,M+1POIØ1588
C              3      F(THETA(I),PS)    F(THETA(I),PF)                  POIØ1589
C              4      F(THETA(I),PS)    U(THETA(I),PF)                  POIØ159Ø
C                                                                       POIØ1591
C       F MUST BE DIMENSIONED AT LEAST (M+1)*(N+1).                     POIØ1592
C                                                                       POIØ1593
C       NOTE                                                            POIØ1594
C                                                                       POIØ1595
C       IF THE TABLE CALLS FOR BOTH THE SOLUTION U AND THE RIGHT SIDE F POIØ1596
C       AT  A CORNER THEN THE SOLUTION MUST BE SPECIFIED.               POIØ1597
C                                                                       POIØ1598
C     IDIMF                                                             POIØ1599
C       THE ROW (OR FIRST) DIMENSION OF THE ARRAY F AS IT APPEARS IN THEPOIØ16ØØ
C       PROGRAM CALLING PWSSSP.  THIS PARAMETER IS USED TO SPECIFY THE  POIØ16Ø1
C       VARIABLE DIMENSION OF F.  IDIMF MUST BE AT LEAST M+1  .         POIØ16Ø2
C                                                                       POIØ16Ø3
C     W                                                                 POIØ16Ø4
C       A ONE-DIMENSIONAL ARRAY THAT MUST BE PROVIDED BY THE USER FOR   POIØ16Ø5
C       WORK SPACE.  THE LENGTH OF W MUST BE AT LEAST 11(M+1)+6(N+1).   POIØ16Ø6
C                                                                       POIØ16Ø7
C                                                                       POIØ16Ø8
C     * * * * * * * * * *     ON OUTPUT     * * * * * * * * * *         POIØ16Ø9
C                                                                       POIØ161Ø
C     F                                                                 POIØ1611
C       CONTAINS THE SOLUTION U(I,J) OF THE FINITE DIFFERENCE           POIØ1612
C       APPROXIMATION FOR THE GRID POINT (THETA(I),PHI(J)),             POIØ1613
C       I = 1,2,...,M+1,   J = 1,2,...,N+1  .                           POIØ1614
C                                                                       POIØ1615
C     PERTRB                                                            POIØ1616
C       IF ONE SPECIFIES A COMBINATION OF PERIODIC, DERIVATIVE OR       POIØ1617
C       UNSPECIFIED BOUNDARY CONDITIONS FOR A POISSON EQUATION          POIØ1618
C       (LAMBDA = Ø), A SOLUTION MAY NOT EXIST.  PERTRB IS A CONSTANT,  POIØ1619
C       CALCULATED AND SUBTRACTED FROM F, WHICH ENSURES THAT A SOLUTION POIØ162Ø
C       EXISTS.  PWSSSP THEN COMPUTES THIS SOLUTION, WHICH IS A LEAST   POIØ1621
C       SQUARES SOLUTION TO THE ORIGINAL APPROXIMATION.  THIS SOLUTION  POIØ1622
C       IS NOT UNIQUE AND IS UNNORMALIZED. THE VALUE OF PERTRB SHOULD   POIØ1623
C       BE SMALL COMPARED TO THE RIGHT SIDE F. OTHERWISE , A SOLUTION   POIØ1624
C       IS OBTAINED TO AN ESSENTIALLY DIFFERENT PROBLEM. THIS COMPARISONPOIØ1625
```

```
C       SHOULD ALWAYS BE MADE TO INSURE THAT A MEANINGFUL SOLUTION HAS  POIØ1626
C       BEEN OBTAINED                                                   POIØ1627
C                                                                       POIØ1628
C     IERROR                                                            POIØ1629
C       AN ERROR FLAG THAT INDICATES INVALID INPUT PARAMETERS.  EXCEPT  POIØ163Ø
C       FOR NUMBERS Ø AND 8, A SOLUTION IS NOT ATTEMPTED.               POIØ1631
C                                                                       POIØ1632
C       = Ø  NO ERROR.                                                  POIØ1633
C       = 1  TS .LT. Ø  OR  TF .GT. PI                                  POIØ1634
C       = 2  TS .GE. TF.                                                POIØ1635
C       = 3  MBDCND .LT. 1 OR MBDCND .GT. 9  .                          POIØ1636
C       = 4  PS .GE. PF.                                                POIØ1637
C       = 5  N IS NOT OF THE FORM (2**P)(3**Q)(5**R) OR N .LE. 2  .     POIØ1638
C       = 6  NBDCND .LT. Ø OR NBDCND .GT. 4  .                          POIØ1639
C       = 7  AN NBDCND OF 1,2, OR 4 IS USED WITH AN                     POIØ164Ø
C            MBDCND OF 5,6,7,8, OR 9  .                                 POIØ1641
C       = 8  ELMBDA .GT. Ø  .                                           POIØ1642
C       = 9  IDIMF .LT. M+1  .                                          POIØ1643
C       = 1Ø TS=Ø AND MBDCND=3,4 OR 8  OR  TF=PI AND MBDCND=2,3 OR 6    POIØ1644
C                                                                       POIØ1645
C       SINCE THIS IS THE ONLY MEANS OF INDICATING A POSSIBLY INCORRECT POIØ1646
C       CALL TO PWSSSP, THE USER SHOULD TEST IERROR AFTER A CALL.       POIØ1647
C                                                                       POIØ1648
C     W                                                                 POIØ1649
C       CONTAINS INTERMEDIATE VALUES THAT MUST NOT BE DESTROYED IF      POIØ165Ø
C       PWSSSP WILL BE CALLED AGAIN WITH INTL = 1  .                    POIØ1651
```

CERTIFICATION OF ALGORITHM 541

Efficient Fortran Subprograms for the Solution of Separable Elliptic Partial Differential Equations [D3]
[P.N. Swarztrauber and R.A. Sweet, *ACM Trans. Math. Software 5*, 3 (September 1979), 352-364.]

Michael Steuerwalt [Recd 10 Feb. 1977 and 31 Jan. 1979]
Los Alamos Scientific Laboratory, P.O. Box 1663, Los Alamos, NM 87545

INTRODUCTION

The problem of computing solutions to modified Helmholtz equations (or, more generally, separable linear elliptic equations) with simple boundary conditions on a rectangle in any of several coordinate frames arises frequently in applications and as an intermediate step in the solution of nonlinear and evolution problems. The cost of developing, documenting, and testing a package for this task is not negligible. Therefore several federal laboratories that can profitably use the National Center for Atmospheric Research (NCAR) package [2] chose to collaborate in its certification.

THE PACKAGE

Physically, the NCAR package consists of about 4700 lines of Fortran code (almost 40 percent are comments), 800 lines of example drivers, and 140 pages of documentation. The package cost about $300,000 to develop. In comparison, EISPACK has 11,500 lines of code (49 percent are comments), 10,000 lines of example drivers, 551 pages of documentation [1], and cost about $900,000.

The heart of the NCAR package is the two routines POIS and BLKTRI, which solve the linear systems arising from standard second-order finite-difference approximations of separable elliptic boundary value problems on rectangles. The package also includes five drivers that build the discrete system from the least possible information: the differential equation and boundary conditions, the geometric region, and the number of (evenly spaced) mesh points in each direction. Most users will communicate with the core routines only through these drivers.

THE CERTIFICATION EFFORT

Each of the five laboratories that had agreed to collaborate in the certification effort assumed responsibility for a particular driver:

PWSCRT: Air Force Weapons Laboratory, Kirtland Air Force Base, Albuquerque, N. Mex.
PWSPLR: Lawrence Livermore Laboratory, Livermore, Calif.
PWSCYL: Sandia Laboratories, Livermore, Calif.
PWSCSP: Sandia Laboratories, Albuquerque, N. Mex.
PWSSSP: Los Alamos Scientific Laboratory, Los Alamos, N. Mex.

Each laboratory agreed to

(a) Compile the entire package.
(b) Verify the results of the seven NCAR example programs.
(c) Verify the correct working of the input error detection code.
(d) For its particular driver, run a test problem using
  (1) several permissible regions,
  (2) several mesh sizes in each direction,
  (3) all possible boundary conditions,
  (4) zero and nonzero values of λ.
(e) Evaluate the documentation.

It is important to note what we did *not* try to do:

(a) Explicitly test the core routines POIS and BLKTRI. Access to these routines was through the drivers alone.

(b) Make efficiency tests. The methods of the core routines are among the best direct methods available, but neither the authors nor the certifiers claim that there are no methods more efficient. In particular, we expect that higher-order discretization schemes would be more efficient for a given (small enough) accuracy.

(c) Make severe tests of the package's robustness.

The schedule outlined above entails a considerable effort. Among the five laboratories there are perhaps 10 different Fortran compilers, so the simple compilation of the package is a good test of its portability. The testing implied by (d) is substantial. For example, the driver PWSSSP admits 9 different possible boundary conditions in the θ direction and 5 in the ϕ direction; not all combinations are compatible, and some are valid only for certain geometries. To complete part (d) for the PWSSSP routine, 8 different regions were used with all possible valid boundary conditions, and with 5 different mesh sizes in the θ direction and 4 in the ϕ direction, for a total of 2360 runs per value of λ. Testing of the other drivers was similar. See the Appendix for details of the test problems.

RESULTS OF THE CERTIFICATION EFFORT

After initial tests, the certification team suggested several changes in the package and the documentation. All our recommendations were adopted. The most significant of these changes are the following.

(1) Version numbers for the package. The version we are certifying here is Version 2.

(2) Correction of initialization errors. The core routines POIS and BLKTRI perform some preliminary computations that need not be repeated if certain problem parameters remain unchanged. Our first tests had revealed program errors that could be avoided only by reinitializing every problem.

(3) Additional code to check for illegal combinations of problem parameters. The documentation clearly indicates that certain choices of boundary conditions are incompatible with particular geometries. For instance, in PWSSSP, three possible choices of the boundary condition at the final value TF of θ require that $TF = \pi$. A sample problem was run with these choices of boundary condition but with $TF < \pi$. The package did not check for such illegal combinations, but simply computed, producing wrong answers. In Version 2, the drivers will also set an error flag if, say, $|TF - \pi|$ is "too large." We feel that the package's robustness

is well enhanced by including this check, which has been done in a portable way and at little computing expense. On the other hand, we should remark that Swarztrauber and Sweet have indicated to us some uneasiness regarding the introduction of the imprecise and machine-dependent notion of "too large," and that no package, whatever its robustness or quality of documentation may be, can protect a user bent on self-immolation.

(4) An extensive list of errata and changes to the documentation. The documentation is neither so exhaustive as the EISPACK guide nor so rich in examples of the routines' use. This is no handicap: use of the NCAR routines is straightforward, whereas EISPACK often provides several options for doing a computation.

All the program tests were repeated for Version 2 to verify the corrections. The tests show that the methods are indeed of second-order accuracy. We deem the package to be valuable software of good quality. We especially commend its design, which permits users to communicate with it in familiar terms so that they do not have to grasp the mechanics of the discretization procedure. We believe the documentation will help that user who only wants answers to his problems to get those answers while remaining in blissful ignorance of details peripheral to his interests.

APPENDIX. DETAILS OF THE TEST PROBLEMS

Each driver may distinguish types of regions according to the geometry of the problem. Some combinations of boundary conditions and regions are not admissible. In this Appendix we tabulate the valid combinations for each driver; the entry "—" indicates no legal combination. We also list the intervals whose Cartesian products form the regions used in the certification tests, and give the true solution u of the various boundary value problems. The appropriate boundary values and the function f can be determined for a given problem from u, λ, and the region. For all problems the values of λ, M, and N used were

λ :	0.0	−1.0			
M :	9	18	36	72	144
N :	15	30	45	60	

PWSCRT

All 25 possible combinations of MBDCND and NBDCND are valid. The two regions used were

$$[A, B] = [-1/2, 1/2]$$

$$[C, D] = [-1/2, 1/2] \qquad [0, 2].$$

The true solution was

$$u = \sin 2\pi(x + 1/8) \cos 2\pi(y + 1/8).$$

PWSPLR

This driver distinguishes two types of regions:

$0 < A$	P
$0 = A$	Z

The compatibility table has 34 entries:

MBDCND	NBDCND 0	1	2	3	4
1	PZ	PZ	PZ	PZ	PZ
2	PZ	PZ	PZ	PZ	PZ
3	P	P	P	P	P
4	P	P	P	P	P
5	Z	—	—	Z	—
6	Z	—	—	Z	—

The four regions used were

$$[A, B] = [1/4, 1] \qquad [0, 1]$$
$$[C, D] = [0, 2\pi/3] \qquad [2\pi, 8\pi/3].$$

The true solution was

$$u = r^3 \cos(4\pi\theta/[D - C] + \pi/4).$$

PWSCYL

This driver distinguishes two types of regions:

$0 < A$	P
$0 = A$	Z

The compatibility table has 40 entries:

	NBDCND				
MBDCND	0	1	2	3	4
1	PZ	PZ	PZ	PZ	PZ
2	PZ	PZ	PZ	PZ	PZ
3	P	P	P	P	P
4	P	P	P	P	P
5	Z	Z	Z	Z	Z
6	Z	Z	Z	Z	Z

The two regions used were:

$$[A, B] = [1, 2] \qquad [0, 1]$$
$$[C, D] = [0, 2\pi].$$

The true solution was

$$u = r^2\cos(z + \pi/4).$$

PWSCSP

This driver recognizes eight types of regions:

	$0 < RS$	$0 = RS$
$0 < TS, TF < \pi$	A	E
$0 = TS, TF < \pi$	B	F
$0 < TS, TF = \pi$	C	G
$0 = TS, TF = \pi$	D	H

The compatibility table has 72 entries:

	NBDCND					
MBDCND	1	2	3	4	5	6
1	ABCD	ABCD	ABCD	ABCD	—	—
2	AB	AB	AB	AB	—	—
3	A	A	A	A	E	E
4	AC	AC	AC	AC	—	—
5	BD	BD	BD	BD	—	—
6	B	B	B	B	F	F
7	CD	CD	CD	CD	—	—
8	C	C	C	C	G	G
9	D	D	D	D	H	H

The eight regions used were

$$[TS,TF] = [\pi/4, \pi/2] \qquad [0, \pi/2] \qquad [\pi/2, \pi] \qquad [0, \pi]$$
$$[RS, RF] = [1, 2] \qquad [0, 1].$$

The true solution was

$$u = r^4\cos^4\theta.$$

PWSSSP

This driver distinguishes four types of regions:

$0 < TS, TF < \pi$	A
$0 = TS, TF < \pi$	B
$0 < TS, TF = \pi$	C
$0 = TS, TF = \pi$	D

There are 59 entries in the compatibility table:

MBDCND	NBDCND 0	1	2	3	4
1	ABCD	ABCD	ABCD	ABCD	ABCD
2	AB	AB	AB	AB	AB
3	A	A	A	A	A
4	AC	AC	AC	AC	AC
5	BD	—	—	BD	—
6	B	—	—	B	—
7	CD	—	—	CD	—
8	C	—	—	C	—
9	D	—	—	D	—

The tests used eight regions:

$[TS, TF] = [\pi/4, \pi/2]$ $[0, \pi/2]$ $[\pi/2, \pi]$ $[0, \pi]$
$[PS, PF] = [\pi/2, 3\pi/2]$ $[0, 2\pi]$.

The true solution was

$$u = \sin^2 \theta \cos 2\phi.$$

Table I. PWSCRT Test
(Region = [−0.5,0.5] × [−0.5,0.5], λ = 0.0. Times are in milliseconds.)

M	N	MBC	NBC	Time		Error
9	15	0	0	8	0.224E−13	0.416E−01
18	30	0	0	32	0.120E−13	0.104E−01
36	45	0	0	96	0.971E−15	0.312E−02
72	60	0	0	271	0.205E−13	0.116E−02
9	15	0	1	6	0.0	0.299E−01
18	30	0	1	23	0.0	0.748E−02
36	45	0	1	71	0.0	0.224E−02
72	60	0	1	198	0.0	0.838E−03
9	15	1	0	6	0.0	0.299E−01
18	30	1	0	22	0.0	0.743E−02
36	45	1	0	65	0.0	0.226E−02
72	60	1	0	178	0.0	0.837E−03
9	15	1	1	4	0.0	0.346E−01
18	30	1	1	16	0.0	0.870E−02
36	45	1	1	49	0.0	0.263E−02
72	60	1	1	136	0.0	0.977E−03

Table II. PWSSSP Test
(Region = $[0,\pi/2] \times [0,2\pi]$, $\lambda = 0.0$. Times are in milliseconds.)

M	N	MBC	NBC	Time	PERTRB	Error
9	15	1	0	6	0.0	0.225E−01
18	30	1	0	22	0.0	0.572E−02
36	45	1	0	66	0.0	0.256E−02
72	60	1	0	180	0.0	0.145E−02
9	15	1	1	4	0.0	0.246E−01
18	30	1	1	16	0.0	0.631E−02
36	45	1	1	50	0.0	0.286E−02
72	60	2	1	137	0.0	0.162E−02
9	15	2	0	6	0.0	0.471E−01
18	30	2	0	23	0.0	0.120E−01
36	45	2	0	67	0.0	0.519E−02
72	60	2	0	181	0.0	0.287E−02
9	15	2	1	5	0.0	0.669E−01
18	30	2	1	17	0.0	0.170E−01
36	45	2	1	51	0.0	0.747E−02
72	60	2	1	138	0.0	0.414E−02

We close with some remarks about the accuracy and speed of the package. Tables I and II summarize some of the test results obtained on a 7600 at Los Alamos, using a local compiler similar to the FTN compiler (optimization level 2). The CPU times, given in milliseconds, reflect the influences of boundary condition and mesh size and are in very good agreement with times reported in [2]. Error is measured in the sup norm over the interior of the mesh of (M + 1) × (N + 1) points; the entries indicate that the finite-difference scheme of the package has second-order accuracy. To see this, note that if the scheme is of second order, then the error on the ith mesh is approximately

$$\text{error}_i = \alpha M_i^{-2} + \beta N_i^{-2},$$

where $M_i = 9*2^i$, $N_i = 15*(i + 1)$, $0 \leqq i \leqq 3$. The Cartesian problem of Table I is symmetric about the line $x = y$, and so $\alpha = \beta$. We therefore expect the ratios $\text{error}_{i-1}/\text{error}_i$ to have the values of 4.00, 3.32, 2.69, respectively. For the spherical surface problem of Table II, α is nearly zero and the expected ratios are 4.00, 2.25, 1.78.

These observations—second-order accuracy and good agreement with the timings of [2]—hold true across the entire test set.

ACKNOWLEDGMENTS

This work was done in collaboration with Martin Havens, Air Force Weapons Laboratory, Kirtland Air Force Base, Albuquerque, N. Mex.; Niel Madsen, Lawrence Livermore Laboratory, Livermore, Calif.; Melvin Scott and Alex Treadway, Sandia Laboratories, Albuquerque, N. Mex.; Richard Basinger, Sandia Laboratories, Livermore, Calif.; and Bill Buzbee, Los Alamos Scientific Laboratory, Los Alamos, N. Mex.

REFERENCES

1. Smith, B.T., Boyle, J.M., Dongarra, J.J., Garbow, B.S., Ikebe, Y., Klema, V.C., and Moler, C.B. Matrix Eigensystem Routines—EISPACK Guide, 2nd ed., *Lecture Notes in Comptr. Sci. 6,* Springer-Verlag, New York, 1976.
2. Swarztrauber, P., and Sweet, R. Efficient FORTRAN subprograms for the solution of elliptic partial differential equations. Rep. NCAR-TN/1A-109, National Center for Atmospheric Research, Boulder, Colo., 1975.

542-P 1- 0

ALGORITHM 542
Incomplete Gamma Functions [S14]

WALTER GAUTSCHI
Purdue University

Key Words and Phrases: computation of incomplete gamma functions, Taylor's series, continued fractions
CR Categories: 5.12
Language: Fortran

DESCRIPTION

This algorithm implements the procedure developed in [1].

REFERENCES

1. GAUTSCHI, W. A computational procedure for incomplete gamma functions. *ACM Trans. Math. Software 5,* 4 (Dec. 1979), 466–481.

ALGORITHM

```
      SUBROUTINE GAM(A, X, ACC, G, GSTAR, IFLG, IFLGST)                  1Ø
C     LET  GAMMA(A)  DENOTE THE GAMMA FUNCTION AND  GAM(A,X)  THE        2Ø
C (COMPLEMENTARY) INCOMPLETE GAMMA FUNCTION,                             3Ø
C                                                                        4Ø
C    GAM(A,X)=INTEGRAL FROM T=X TO T=INFINITY OF EXP(-T)*T**(A-1).       5Ø
C                                                                        6Ø
C LET  GAMSTAR(A,X)  DENOTE TRICOMI:S FORM OF THE INCOMPLETE GAMMA       7Ø
C FUNCTION, WHICH FOR A.GT.Ø. IS DEFINED BY                              8Ø
C                                                                        9Ø
C  GAMSTAR(A,X)=(X**(-A)/GAMMA(A))*INTEGRAL FROM T=Ø TO T=X OF           1ØØ
C               EXP(-T)*T**(A-1),                                        11Ø
C                                                                        12Ø
C AND FOR A.LE.Ø. BY ANALYTIC CONTINUATION. FOR THE PURPOSE OF           13Ø
C THIS SUBROUTINE, THESE FUNCTIONS ARE NORMALIZED AS FOLLOWS&            14Ø
C                                                                        15Ø
C            GAM(A,X)/GAMMA(A),  IF A.GT.Ø.,                             16Ø
C                                                                        17Ø
C     G(A,X)=                                                            18Ø
C                                                                        19Ø
C            EXP(X)*X**(-A)*GAM(A,X),  IF A.LE.Ø.,                       2ØØ
C                                                                        21Ø
C     GSTAR(A,X)=(X**A)*GAMSTAR(A,X).                                    22Ø
C                                                                        23Ø
C THE PROGRAM BELOW ATTEMPTS TO EVALUATE  G(A,X)  AND  GSTAR(A,X),       24Ø
C BOTH TO AN ACCURACY OF ACC SIGNIFICANT DECIMAL DIGITS, FOR ARBI-       25Ø
C TRARY REAL VALUES OF  A  AND NONNEGATIVE VALUES OF  X. THE SUB-        26Ø
C ROUTINE AUTOMATICALLY CHECKS FOR UNDERFLOW AND OVERFLOW CONDI-         27Ø
C TIONS AND RETURNS APPROPRIATE WARNINGS THROUGH THE OUTPUT PARA-        28Ø
```

Received 4 April 1977; revised 31 August 1978.

Author's address: Department of Computer Sciences, Purdue University, Mathematical Sciences Building, Room 442, West Lafayette, IN 47907.

```
C METERS  IFLG, IFLGST. A RESULT THAT UNDERFLOWS IS RETURNED WITH          29Ø
C THE VALUE Ø., ONE THAT OVERFLOWS WITH THE VALUE OF THE LARGEST          3ØØ
C MACHINE-REPRESENTABLE NUMBER.                                           31Ø
C                                                                         32Ø
C     NEAR LINES IN THE (A,X)-PLANE, A.LT.Ø., ALONG WHICH  GSTAR          33Ø
C VANISHES, THE ACCURACY SPECIFIED WILL BE ATTAINED ONLY IN TERMS          34Ø
C OF ABSOLUTE ERROR, NOT RELATIVE ERROR. THERE ARE OTHER (RARE)           35Ø
C INSTANCES IN WHICH THE ACCURACY ATTAINED IS SOMEWHAT LESS THAN           36Ø
C THE ACCURACY SPECIFIED. THE DISCREPANCY, HOWEVER, SHOULD NEVER          37Ø
C EXCEED ONE OR TWO (DECIMAL) ORDERS OF ACCURACY# NO INDICATION           38Ø
C OF THIS IS GIVEN THROUGH ERROR FLAGS.                                   39Ø
C                                                                         4ØØ
C     PARAMETER LIST&                                                     41Ø
C                                                                         42Ø
C        A - - THE FIRST ARGUMENT OF G AND GSTAR. TYPE REAL.              43Ø
C        X - - THE SECOND ARGUMENT OF G AND GSTAR. TYPE REAL.             44Ø
C      ACC - - THE NUMBER OF CORRECT SIGNIFICANT DECIMAL DIGITS           45Ø
C              DESIRED IN THE RESULTS. TYPE REAL.                         46Ø
C        G - - AN OUTPUT VARIABLE RETURNING THE VALUE OF G(A,X).          47Ø
C              TYPE REAL.                                                 48Ø
C    GSTAR - - AN OUTPUT VARIABLE RETURNING THE VALUE OF                  49Ø
C              GSTAR(A,X). TYPE REAL.                                     5ØØ
C     IFLG - - AN ERROR FLAG INDICATING A NUMBER OF ERROR CONDI-          51Ø
C              TIONS IN G UPON EXECUTION. TYPE INTEGER.                   52Ø
C   IFLGST - - AN ERROR FLAG INDICATING A NUMBER OF ERROR CONDI-          53Ø
C              TIONS IN GSTAR UPON EXECUTION. TYPE INTEGER.               54Ø
C              THE VALUES OF IFLG AND IFLGST HAVE THE FOLLOWING           55Ø
C              MEANINGS&                                                  56Ø
C              Ø - NO ERROR CONDITION.                                    57Ø
C              1 - ILLEGAL NEGATIVE ARGUMENT X. THE ROUTINE EXITS         58Ø
C                  WITH THE VALUES ZERO FOR G AND GSTAR.                  59Ø
C              2 - INFINITELY LARGE RESULT AT X=Ø. THE ROUTINE            6ØØ
C                  RETURNS THE LARGEST MACHINE-REPRESENTABLE NUMBER.      61Ø
C              3 - (ONLY FOR IFLGST) GSTAR IS INDETERMINATE AT            62Ø
C                  A=Ø.· AND X=Ø. THE ROUTINE RETURNS THE VALUE 1.,       63Ø
C                  WHICH IS THE LIMIT VALUE AS X TENDS TO ZERO FOR        64Ø
C                  FIXED A=Ø.                                             65Ø
C              4 - THE RESULT UNDERFLOWS. IT IS SET EQUAL TO Ø.           66Ø
C              5 - THE RESULT OVERFLOWS. IT IS SET EQUAL TO THE           67Ø
C                  LARGEST MACHINE-REPRESENTABLE NUMBER, WITH             68Ø
C                  PROPER SIGN.                                           69Ø
C              6 - CONVERGENCE FAILS WITHIN 6ØØ ITERATIONS, EITHER        7ØØ
C                  IN TAYLOR:S SERIES OR IN LEGENDRE:S CONTINUED          71Ø
C                  FRACTION. REASON UNKNOWN. THE COMPUTATION IS           72Ø
C                  ABORTED AND THE ROUTINE RETURNS WITH ZERO              73Ø
C                  VALUES FOR G AND GSTAR.                                74Ø
C                                                                         75Ø
C     ALL MACHINE-DEPENDENT PARAMETERS ARE COLLECTED IN THE FIRST         76Ø
C DATA DECLARATION. THEY ARE AS FOLLOWS&                                  77Ø
C                                                                         78Ø
C     PREC - - THE SINGLE PRECISION ACCURACY, TO BE SET APPROXI-          79Ø
C              MATELY EQUAL TO BETA*ALOG(2.)/ALOG(1Ø.), WHERE BETA        8ØØ
C              IS THE NUMBER OF BINARY DIGITS AVAILABLE IN THE MAN-       81Ø
C              TISSA OF THE SINGLE PRECISION FLOATING-POINT WORD.         82Ø
C              TYPE REAL.                                                 83Ø
C   TOPEXP - - APPROXIMATELY THE LARGEST POSITIVE NUMBER T SUCH           84Ø
C              THAT 1Ø.**T IS STILL REPRESENTABLE ON THE COMPUTER         85Ø
C              IN SINGLE PRECISION FLOATING-POINT ARITHMETIC.             86Ø
C              TYPE REAL.                                                 87Ø
C   BOTEXP - - APPROXIMATELY THE SMALLEST NEGATIVE NUMBER T SUCH          88Ø
C              THAT 1Ø.**T IS STILL REPRESENTABLE ON THE COMPUTER         89Ø
C              IN SINGLE PRECISION FLOATING-POINT ARITHMETIC.             9ØØ
C              TYPE REAL.                                                 91Ø
C                                                                         92Ø
C IN THE PROGRAM BELOW THESE PARAMETERS ARE SET TO CORRESPOND TO          93Ø
C THE MACHINE CHARACTERISTICS OF THE CDC 65ØØ COMPUTER.                   94Ø
C                                                                         95Ø
C     THE SECOND DATA DECLARATION CONTAINS THE SINGLE PRECISION           96Ø
C VALUE OF ALOG(1Ø.). THE NEXT DATA DECLARATION CONTAINS THE SUCCES-       97Ø
C SIVE COEFFICIENTS IN THE MACLAURIN EXPANSION OF (1/GAMMA(A+1))-1.       98Ø
C THEY ARE GIVEN TO AS MANY DECIMAL PLACES AS IS NECESSARY TO ACHIEVE      99Ø
C MACHINE PRECISION (ON THE CDC 65ØØ COMPUTER) IN THE RANGE              1ØØØ
C ABS(A).LE..5. MORE ACCURATE VALUES OF THESE COEFFICIENTS (TO            1Ø1Ø
C 31 DECIMAL PLACES) CAN BE FOUND IN TABLE 5 OF J.W.WRENCH,JR.,           1Ø2Ø
C ::CONCERNING TWO SERIES FOR THE GAMMA FUNCTION::, MATH. COMPUT.         1Ø3Ø
C 22, 1968, 617-626.                                                      1Ø4Ø
```

```
C                                                                       1Ø5Ø
C     THE SUBROUTINE CALLS ON A FUNCTION SUBROUTINE, NAMED  ALGA,       1Ø6Ø
C WHICH IS TO PROVIDE SINGLE PRECISION VALUES OF THE LOGARITHM OF       1Ø7Ø
C THE GAMMA FUNCTION FOR ARGUMENTS LARGER THAN OR EQUAL TO .5.          1Ø8Ø
C A POSSIBLE VERSION OF SUCH A FUNCTION SUBROUTINE IS APPENDED          1Ø9Ø
C TO THE PRESENT SUBROUTINE. IT IS TAYLORED TO THE ACCURACY RE-         11ØØ
C QUIREMENTS OF THE CDC 65ØØ COMPUTER, AND USES RATIONAL APPROXI-       111Ø
C MATIONS DUE TO CODY AND HILLSTROM (MATH. COMPUT. 21, 1967, 198-       112Ø
C 2Ø3).                                                                 113Ø
C                                                                       114Ø
C     REFERENCE - W. GAUTSCHI, ::A COMPUTATIONAL PROCEDURE FOR          115Ø
C INCOMPLETE GAMMA FUNCTIONS::, ACM TRANS. MATH. SOFTWARE.              116Ø
C                                                                       117Ø
      DIMENSION C(18)                                                   118Ø
      DATA PREC, TOPEXP, BOTEXP /1.445E1,3.22E2,-2.93E2/                119Ø
      DATA AL1Ø /2.3Ø2585Ø92994Ø5EØ/                                    12ØØ
      DATA C /.57721566490153 3EØ,-.65587Ø7152Ø254EØ,                   121Ø
     * -4.2ØØ2635Ø34Ø95E-2,.16653861138229EØ,-4.219773455554E-2,        122Ø
     * -9.62197152788E-3,7.21894324666E-3,-1.1651675919E-3,             123Ø
     * -2.152416741E-4,1.28Ø5Ø282E-4,-2.Ø134855E-5,-1.25Ø49E-6,         124Ø
     * 1.133Ø3E-6,-2.Ø563E-7,6.12E-9,5.ØØE-9,-1.2E-9,1.E-1Ø/            125Ø
      G = Ø.                                                            126Ø
      GSTAR = Ø.                                                        127Ø
      IF (X.LT.Ø.) GO TO 29Ø                                            128Ø
C                                                                       129Ø
C INITIALIZATION                                                        13ØØ
C                                                                       131Ø
      IFLG = Ø                                                          132Ø
      IFLGST = Ø                                                        133Ø
      I = Ø                                                             134Ø
      IF (X.GT.Ø.) ALX = ALOG(X)                                        135Ø
      ALPHA = X + .25                                                   136Ø
      IF (X.LT..25 .AND. X.GT.Ø.) ALPHA = ALOG(.5)/ALX                  137Ø
      ALPREC = AL1Ø*PREC                                                138Ø
      TOP = AL1Ø*TOPEXP                                                 139Ø
      BOT = AL1Ø*BOTEXP                                                 14ØØ
      AINF = 1Ø.**TOPEXP                                                141Ø
      EPS = .5*1Ø.**(-ACC)                                              142Ø
      EPS1 = EPS/1ØØ.                                                   143Ø
      SGA = 1.                                                          144Ø
      IF (A.LT.Ø.) SGA = -SGA                                           145Ø
      AE = A                                                            146Ø
      AA = ABS(A)                                                       147Ø
      AP1 = A + 1.                                                      148Ø
      AEP1 = AP1                                                        149Ø
      MA = .5 - A                                                       15ØØ
      FMA = FLOAT(MA)                                                   151Ø
      AEPS = A + FMA                                                    152Ø
      SGAE = 1.                                                         153Ø
      IF (AEPS.LT.Ø.) SGAE = -SGAE                                      154Ø
      AAEPS = ABS(AEPS)                                                 155Ø
      ALGP1 = Ø.                                                        156Ø
C                                                                       157Ø
C EVALUATION OF THE LOGARITHM OF THE ABSOLUTE VALUE OF                  158Ø
C GAMMA(A+1.) AND DETERMINATION OF THE SIGN OF GAMMA(A+1.)              159Ø
C                                                                       16ØØ
      SGGA = 1.                                                         161Ø
      IF (MA.LE.Ø) GO TO 1Ø                                             162Ø
      IF (AEPS.EQ.Ø.) GO TO 2Ø                                          163Ø
      SGGA = SGAE                                                       164Ø
      IF (MA.EQ.2*(MA/2)) SGGA = -SGGA                                  165Ø
      ALGP1 = ALGA(AEPS+1.) - ALOG(AAEPS)                               166Ø
      IF (MA.EQ.1) GO TO 2Ø                                             167Ø
      ALGP1 = ALGP1 + ALGA(1.-AEPS) - ALGA(FMA-AEPS)                    168Ø
      GO TO 2Ø                                                          169Ø
   1Ø ALGP1 = ALGA(AP1)                                                 17ØØ
   2Ø ALGEP1 = ALGP1                                                    171Ø
      IF (X.GT.Ø.) GO TO 6Ø                                             172Ø
C                                                                       173Ø
C EVALUATION OF GSTAR(A,Ø.) AND G(A,Ø.)                                 174Ø
C                                                                       175Ø
      IF (A) 3Ø, 4Ø, 5Ø                                                 176Ø
   3Ø IFLGST = 2                                                        177Ø
      GSTAR = AINF                                                      178Ø
      G = 1./AA                                                         179Ø
      RETURN                                                            18ØØ
```

```
   4Ø IFLGST = 3                                                        181Ø
      GSTAR = 1.                                                        182Ø
      IFLG = 2                                                          183Ø
      G = AINF                                                          184Ø
      RETURN                                                            185Ø
   5Ø G = 1.                                                            186Ø
      RETURN                                                            187Ø
   6Ø IF (A.GT.ALPHA) GO TO 22Ø                                         188Ø
      IF (X.GT.1.5) GO TO 24Ø                                           189Ø
      IF (A.LT.-.5) GO TO 17Ø                                           19ØØ
C                                                                       191Ø
C DIRECT EVALUATION OF G(A,X) AND GSTAR(A,X) FOR X.LE.1.5               192Ø
C AND -.5.LE.A.LE.ALPHA(X)                                              193Ø
C                                                                       194Ø
      GSTAR = 1.                                                        195Ø
    IF (A.GE..5) GO TO 11Ø                                              196Ø
  7Ø SUM = C(18)                                                        197Ø
    DO 8Ø K=1,17                                                        198Ø
      K1 = 18 - K                                                       199Ø
      SUM = AE*SUM + C(K1)                                              2ØØØ
  8Ø CONTINUE                                                           2Ø1Ø
    GA = -SUM/(1.+AE*SUM)                                               2Ø2Ø
    Y = AE*ALX                                                          2Ø3Ø
    IF (ABS(Y).GE.1.) GO TO 1ØØ                                         2Ø4Ø
    SUM = 1.                                                            2Ø5Ø
    TERM = 1.                                                           2Ø6Ø
    K = 1                                                               2Ø7Ø
  9Ø K = K + 1                                                          2Ø8Ø
    IF (K.GT.6ØØ) GO TO 33Ø                                             2Ø9Ø
    TERM = Y*TERM/FLOAT(K)                                              21ØØ
    SUM = SUM + TERM                                                    211Ø
    IF (ABS(TERM).GT.EPS1*SUM) GO TO 9Ø                                 212Ø
    U = GA - SUM*ALX                                                    213Ø
    GO TO 12Ø                                                           214Ø
 1ØØ U = GA - (EXP(Y)-1.)/AE                                            215Ø
    GO TO 12Ø                                                           216Ø
 11Ø U = EXP(ALGA(A)) - (X**A)/A                                        217Ø
 12Ø P = AE*X                                                           218Ø
    Q = AEP1                                                            219Ø
    R = AE + 3.                                                         22ØØ
    TERM = 1.                                                           221Ø
    SUM = 1.                                                            222Ø
    K = 1                                                               223Ø
 13Ø K = K + 1                                                          224Ø
    IF (K.GT.6ØØ) GO TO 33Ø                                             225Ø
    P = P + X                                                           226Ø
    Q = Q + R                                                           227Ø
    R = R + 2.                                                          228Ø
    TERM = -P*TERM/Q                                                    229Ø
    SUM = SUM + TERM                                                    23ØØ
    IF (ABS(TERM).GT.EPS1*SUM) GO TO 13Ø                                231Ø
    V = (X**AEP1)*SUM/AEP1                                              232Ø
    G = U + V                                                           233Ø
    IF (I.EQ.1) GO TO 18Ø                                               234Ø
    IF (A) 14Ø, 15Ø, 16Ø                                                235Ø
 14Ø T = EXP(X)*X**(-A)                                                 236Ø
    G = T*G                                                             237Ø
    GSTAR = 1. - A*G*EXP(-ALGP1)/T                                      238Ø
    RETURN                                                              239Ø
 15Ø G = EXP(X)*G                                                       24ØØ
      RETURN                                                            241Ø
  16Ø G = A*G*EXP(-ALGP1)                                               242Ø
      GSTAR = 1. - G                                                    243Ø
      RETURN                                                            244Ø
C                                                                       245Ø
C RECURSIVE EVALUATION OF G(A,X) FOR X.LE.1.5 AND A.LT.-.5              246Ø
C                                                                       247Ø
  17Ø I = 1                                                             248Ø
      AE = AEPS                                                         249Ø
      AEP1 = AEPS + 1.                                                  25ØØ
      IF (X.LT..25 .AND. AE.GT.ALPHA) GO TO 21Ø                         251Ø
      GO TO 7Ø                                                          252Ø
  18Ø G = G*EXP(X)*X**(-AE)                                             253Ø
      DO 19Ø K=1,MA                                                     254Ø
        G = (1.-X*G)/(FLOAT(K)-AE)                                      255Ø
```

```
  19Ø CONTINUE                                                          256Ø
      ALG = ALOG(G)                                                     257Ø
C                                                                       258Ø
C EVALUATION OF GSTAR(A,X) IN TERMS OF G(A,X)                           259Ø
C                                                                       26ØØ
  2ØØ GSTAR = 1.                                                        261Ø
      IF (MA.GE.Ø .AND. AEPS.EQ.Ø.) RETURN                             262Ø
      SGT = SGA*SGGA                                                    263Ø
      T = ALOG(AA) - X + A*ALX + ALG - ALGP1                            264Ø
      IF (T.LT.-ALPREC) RETURN                                          265Ø
      IF (T.GE.TOP) GO TO 32Ø                                           266Ø
      GSTAR = 1. - SGT*EXP(T)                                           267Ø
      RETURN                                                            268Ø
  21Ø I = 2                                                             269Ø
      ALGEP1 = ALGA(AEP1)                                               27ØØ
C                                                                       271Ø
C EVALUATION OF GSTAR(A,X) FOR A.GT.ALPHA(X) BY TAYLOR                  272Ø
C EXPANSION                                                             273Ø
C                                                                       274Ø
  22Ø G = 1.                                                            275Ø
      TERM = 1.                                                         276Ø
      SUM = 1.                                                          277Ø
      K = Ø                                                             278Ø
  23Ø K = K + 1                                                         279Ø
      IF (K.GT.6ØØ) GO TO 34Ø                                           28ØØ
      TERM = X*TERM/(AE+FLOAT(K))                                       281Ø
      SUM = SUM + TERM                                                  282Ø
      IF (ABS(TERM).GT.EPS*SUM) GO TO 23Ø                               283Ø
      ALGS = AE*ALX - X + ALOG(SUM) - ALGEP1                            284Ø
      IF (ALGS.LE.BOT) GO TO 31Ø                                        285Ø
      GSTAR = EXP(ALGS)                                                 286Ø
      G = 1. - GSTAR                                                    287Ø
      IF (I.NE.2) RETURN                                                288Ø
      G = G*EXP(ALGEP1)/AE                                              289Ø
      GO TO 18Ø                                                         29ØØ
C                                                                       291Ø
C EVALUATION OF G(A,X) FOR X.GT.1.5 AND A.LE.ALPHA(X) BY                292Ø
C MEANS OF THE LEGENDRE CONTINUED FRACTION                              293Ø
C                                                                       294Ø
  24Ø GSTAR = 1.                                                        295Ø
      XPA = X + 1. - A                                                  296Ø
      XMA = X - 1. - A                                                  297Ø
      P = Ø.                                                            298Ø
      Q = XPA*XMA                                                       299Ø
      R = 4.*XPA                                                        3ØØØ
    S = -A + 1.                                                         3Ø1Ø
    TERM = 1.                                                           3Ø2Ø
    SUM = 1.                                                            3Ø3Ø
    RHO = Ø.                                                            3Ø4Ø
    K = 1                                                               3Ø5Ø
25Ø K = K + 1                                                           3Ø6Ø
    IF (K.GT.6ØØ) GO TO 33Ø                                             3Ø7Ø
    P = P + S                                                           3Ø8Ø
    Q = Q + R                                                           3Ø9Ø
    R = R + 8.                                                          31ØØ
    S = S + 2.                                                          311Ø
    T = P*(1.+RHO)                                                      312Ø
    RHO = T/(Q-T)                                                       313Ø
    TERM = RHO*TERM                                                     314Ø
    SUM = SUM + TERM                                                    315Ø
    IF (ABS(TERM).GT.EPS*SUM) GO TO 25Ø                                 316Ø
    IF (A) 26Ø, 27Ø, 28Ø                                                317Ø
26Ø G = SUM/XPA                                                         318Ø
    ALG = ALOG(G)                                                       319Ø
    GO TO 2ØØ                                                           32ØØ
27Ø G = SUM/XPA                                                         321Ø
    RETURN                                                              322Ø
28Ø ALG = A*ALX - X + ALOG(A*SUM/XPA) - ALGP1                           323Ø
    IF (ALG.LE.BOT) GO TO 3ØØ                                           324Ø
    G = EXP(ALG)                                                        325Ø
    GSTAR = 1. - G                                                      326Ø
    RETURN                                                              327Ø
29Ø IFLG = 1                                                            328Ø
    IFLGST = 1                                                          329Ø
    RETURN                                                              33ØØ
3ØØ IFLG = 4                                                            331Ø
```

```
      RETURN                                                            332Ø
  31Ø IFLGST = 4                                                        333Ø
      RETURN                                                            334Ø
  32Ø IFLGST = 5                                                        335Ø
      GSTAR = -SGT*AINF                                                 336Ø
      RETURN                                                            337Ø
  33Ø IFLG = 6                                                          338Ø
      RETURN                                                            339Ø
  34Ø IFLGST = 6                                                        34ØØ
      RETURN                                                            341Ø
      END                                                               342Ø

      FUNCTION ALGA(X)                                                  343Ø
      DIMENSION CNUM(8), CDEN(8)                                        344Ø
      DATA CNUM /4.12Ø843185847770,85.6898206283131 7,243.175243524421,  345Ø
     * -261.7218583856145,-922.2613728801522,-517.6383498023218,         346Ø
     * -77.4106407133295 3,-2.208843997216182/, CDEN                     347Ø
     * /1.,45.64677187585908,377.8372484823942,951.32359767970 6,        348Ø
     * 846.0755362020782,262.3083470269460,24.4351966250 6312,           349Ø
     * .4097792921092615/                                                35ØØ
      XI = AINT(X)                                                      351Ø
      IF (X-XI.GT..5) XI = XI + 1.                                      352Ø
      M = IFIX(XI) - 1                                                  353Ø
      XE = X                                                            354Ø
      IF (M.EQ.-1) XE = X + 1.                                          355Ø
      IF (M.GT.Ø) XE = X - FLOAT(M)                                     356Ø
      SNUM = CNUM(1)                                                    357Ø
      SDEN = CDEN(1)                                                    358Ø
      DO 1Ø K=2,8                                                       359Ø
        SNUM = XE*SNUM + CNUM(K)                                        36ØØ
          SDEN = XE*SDEN + CDEN(K)                                      361Ø
    1Ø CONTINUE                                                         362Ø
        ALGA = (XE-1.)*SNUM/SDEN                                        363Ø
        IF (M.GT.-1) GO TO 2Ø                                           364Ø
        ALGA = ALGA - ALOG(X)                                           365Ø
        RETURN                                                          366Ø
    2Ø IF (M.EQ.Ø) RETURN                                               367Ø
        P = XE                                                          368Ø
        IF (M.EQ.1) GO TO 4Ø                                            369Ø
        MM1 = M - 1                                                     37ØØ
C                                                                       371Ø
C THE NEXT STATEMENT IS DESIGNED TO AVOID POSSIBLE OVERFLOW IN THE      372Ø
C COMPUTATION OF P. THE CONDITION IN THE IF-CLAUSE EXPRESSES THE        373Ø
C INEQUALITY 1*3*5* ... *(2*M+1)/(2**M).GE.Q, WHERE Q IS THE LARGEST    374Ø
C MACHINE-REPRESENTABLE NUMBER.                                         375Ø
C                                                                       376Ø
        IF (M.GE.176) GO TO 5Ø                                          377Ø
        DO 3Ø K=1,MM1                                                   378Ø
          P = (XE+FLOAT(K))*P                                           379Ø
    3Ø CONTINUE                                                         38ØØ
    4Ø ALGA = ALGA + ALOG(P)                                            381Ø
        RETURN                                                          382Ø
    5Ø ALGA = ALGA + ALOG(XE)                                           383Ø
        DO 6Ø K=1,MM1                                                   384Ø
          ALGA = ALGA + ALOG(XE+FLOAT(K))                               385Ø
    6Ø CONTINUE                                                         386Ø
        RETURN                                                          387Ø
        END                                                             388Ø
C                                                                       389Ø
C DRIVER1 - ERROR FUNCTIONS                                             39ØØ
C                                                                       391Ø
C ERF X  FOR  X=Ø(.Ø5)1.5  IN SINGLE AND DOUBLE PRECISION WITH          392Ø
C RELATIVE ERROR. CHECK AGAINST TABLE 7.1 IN NBS HANDBOOK.              393Ø
C                                                                       394Ø
      DOUBLE PRECISION DPI, DC, DX, DXSQ, DG, DGSTAR, DERF, DXMSQ, DERFC 395Ø
      PI = 4.*ATAN(1.)                                                  396Ø
      DPI = 4.DØ*DATAN(1.DØ)                                            397Ø
      WRITE (6,99999)                                                   398Ø
      DO 1Ø I=1,31                                                      399Ø
        X = FLOAT(I-1)*.Ø5                                              4ØØØ
        DX = DBLE(FLOAT(I-1))*5.D-2                                     4Ø1Ø
        XSQ = X*X                                                       4Ø2Ø
        DXSQ = DX*DX                                                    4Ø3Ø
        CALL GAM(.5, XSQ, 8., G, ERF, IFLG, IFLGST)                     4Ø4Ø
        CALL DGAM(.5DØ, DXSQ, 16., DG, DERF, IFGD, IFGSTD)              4Ø5Ø
```

```
      ERROR = SNGL(DABS(DBLE(ERF)-DERF))                                4Ø6Ø
      IF (DERF.NE.Ø.DØ) ERROR = ABS(ERROR/SNGL(DERF))                   4Ø7Ø
      WRITE (6,99998) X, ERF, DERF, ERROR, IFLG, IFLGST, IFGD, IFGSTD   4Ø8Ø
   1Ø CONTINUE                                                          4Ø9Ø
C                                                                       41ØØ
C X*EXP(X*X)*ERFC X  FOR X**(-2)=.ØØ5(.ØØ5).25  IN SINGLE AND          411Ø
C DOUBLE PRECISION WITH RELATIVE ERROR. CHECK AGAINST TABLE 7.3         412Ø
C IN NBS HANDBOOK.                                                      413Ø
C                                                                       414Ø
      WRITE (6,99997)                                                   415Ø
      DO 3Ø I=1,5Ø                                                      416Ø
        XMSQ = FLOAT(I)*.ØØ5                                            417Ø
        DXMSQ = DBLE(FLOAT(I))*5.D-3                                    418Ø
        XSQ = 1./XMSQ                                                   419Ø
        DXSQ = 1.DØ/DXMSQ                                               42ØØ
        CALL GAM(.5, XSQ, 7., G, GSTAR, IFLG, IFLGST)                   421Ø
        CALL DGAM(.5DØ, DXSQ, 16., DG, DGSTAR, IFGD, IFGSTD)            422Ø
        ERFC = Ø.                                                       423Ø
        DERFC = Ø.DØ                                                    424Ø
        IF (XSQ.GT.74Ø.) GO TO 2Ø                                       425Ø
        ERFC = SQRT(XSQ)*EXP(XSQ)*G                                     426Ø
        DERFC = DSQRT(DXSQ)*DEXP(DXSQ)*DG                               427Ø
   2Ø   ERROR = SNGL(DABS(DBLE(ERFC)-DERFC))                            428Ø
        IF (DERFC.NE.Ø.DØ) ERROR = ABS(ERROR/SNGL(DERFC))               429Ø
        WRITE (6,99996) XMSQ, ERFC, DERFC, ERROR, IFLG, IFLGST, IFGD,   43ØØ
     *   IFGSTD                                                         431Ø
   3Ø CONTINUE                                                          432Ø
C                                                                       433Ø
C ERFC(SQRT(N*PI)) FOR N=1(1)1Ø IN SINGLE AND DOUBLE PRECISION          434Ø
C WITH RELATIVE ERROR. CHECK AGAINST TABLE 7.3 IN NBS HANDBOOK.         435Ø
C                                                                       436Ø
      WRITE (6,99995)                                                   437Ø
      DO 4Ø N=1,1Ø                                                      438Ø
        X = FLOAT(N)*PI                                                 439Ø
        DX = DBLE(FLOAT(N))*DPI                                         44ØØ
        CALL GAM(.5, X, 8., ERFC, GSTAR, IFLG, IFLGST)                  441Ø
        CALL DGAM(.5DØ, DX, 16., DERFC, DGSTAR, IFGD, IFGSTD)           442Ø
        ERROR = ABS(SNGL((DBLE(ERFC)-DERFC)/DERFC))                     443Ø
        WRITE (6,99994) N, ERFC, DERFC, ERROR, IFLG, IFLGST, IFGD,      444Ø
     *   IFGSTD                                                         445Ø
   4Ø CONTINUE                                                          446Ø
      STOP                                                              447Ø
99999 FORMAT (/26X, 1HX, 8X, 5HERF X, 14X, 5HERF X, 12X, 5HERROR, 4X,   448Ø
     * 4HIFLG, 1X, 6HIFLGST, 2X, 4HIFGD, 1X, 6HIFGSTD/)                 449Ø
99998 FORMAT (2ØX, E1Ø.2, E15.7, D23.15, E1Ø.2, 4I6)                    45ØØ
99997 FORMAT (//23X, 7HX**(-2), 1ØX, 17HX*EXP(X*X)*ERFC X, 13X, 5HERROR, 451Ø
     * 4X, 4HIFLG, 1X, 6HIFLGST, 2X, 4HIFGD, 1X, 6HIFGSTD/)             452Ø
99996 FORMAT (2ØX, E1Ø.2, E14.6, D23.15, E1Ø.2, 4I6)                    453Ø
99995 FORMAT (//27X, 1HN, 13X, 16HERFC(SQRT(N*PI)), 14X, 5HERROR, 4X,   454Ø
     * 4HIFLG, 1X, 6HIFLGST, 2X, 4HIFGD, 1X, 6HIFGSTD/)                 455Ø
99994 FORMAT (I28, E17.7, D23.15, E1Ø.2, 4I6)                           456Ø
      END                                                               457Ø
C                                                                       458Ø
C DRIVER3 - EXPONENTIAL INTEGRAL                                        459Ø
C                                                                       46ØØ
C ESUBN(X)  FOR  N=Ø(1)2Ø, X=VAR,  IN SINGLE AND DOUBLE PRECISION       461Ø
C                                                                       462Ø
C ESUBN(X)  FOR  N=Ø(1)2Ø, X=VAR,  IN SINGLE AND DOUBLE PRECISION       463Ø
C WITH RELATIVE ERROR. CHECK AGAINST TABLES I AND II IN PAGUROVA.       464Ø
C                                                                       465Ø
      DOUBLE PRECISION DX1, DX2, DX, DA, DG, DGSTAR, DESUBN, DP, DANU,  466Ø
     * DESBNU, DLGA                                                     467Ø
      DIMENSION X1(1Ø), X2(6), DX1(1Ø), DX2(6)                          468Ø
      DATA X1 /Ø.,.Ø1,.Ø5,.2,.5,1.5,5.1,1Ø.,14.7,19.8/                  469Ø
      DATA X2 /.Ø1,.37,1.44,3.Ø2,6.57,2Ø./                             47ØØ
      DATA DX1 /Ø.DØ,1.D-2,5.D-2,.2DØ,.5DØ,1.5DØ,5.1DØ,1.D1,1.47D1,     471Ø
     * 1.98D1/                                                          472Ø
      DATA DX2 /1.D-2,.37DØ,1.44DØ,3.Ø2DØ,6.57DØ,2.D1/                  473Ø
      WRITE (6,99999)                                                   474Ø
      DO 5Ø I=1,1Ø                                                      475Ø
        X = X1(I)                                                       476Ø
        DX = DX1(I)                                                     477Ø
        DO 4Ø J=1,21                                                    478Ø
          N = J - 1                                                     479Ø
          A = FLOAT(-N+1)                                               48ØØ
```

```
            DA = DBLE(A)                                                    481Ø
            CALL GAM(A, X, 8., G, GSTAR, IFLG, IFLGST)                      482Ø
            CALL DGAM(DA, DX, 16., DG, DGSTAR, IFGD, IFGSTD)                483Ø
            IF (X.GT.Ø.) GO TO 1Ø                                           484Ø
            ESUBN = Ø.                                                      485Ø
            DESUBN = Ø.DØ                                                   486Ø
            IF (N.LE.1) GO TO 3Ø                                            487Ø
            ESUBN = G                                                       488Ø
            DESUBN = DG                                                     489Ø
            GO TO 3Ø                                                        49ØØ
   1Ø       IF (N.NE.Ø) GO TO 2Ø                                            491Ø
            ESUBN = G/X                                                     492Ø
            DESUBN = DG/DX                                                  493Ø
            GO TO 3Ø                                                        494Ø
   2Ø       ESUBN = EXP(-X)*G                                               495Ø
            DESUBN = DEXP(-DX)*DG                                           496Ø
   3Ø       ERROR = SNGL(DABS(DBLE(ESUBN)-DESUBN))                          497Ø
            IF (DESUBN.NE.Ø.DØ) ERROR = ABS(ERROR/SNGL(DESUBN))             498Ø
            WRITE (6,99998) N, A, X, ESUBN, DESUBN, ERROR, IFLG, IFLGST,    499Ø
     *       IFGD, IFGSTD                                                   5ØØØ
   4Ø     CONTINUE                                                         5Ø1Ø
          WRITE (6,99997)                                                  5Ø2Ø
   5Ø CONTINUE                                                             5Ø3Ø
C                                                                          5Ø4Ø
C ESUBNU(X)  FOR  NU=Ø(.1)1., X=VAR,  IN SINGLE AND DOUBLE                 5Ø5Ø
C PRECISION WITH RELATIVE ERROR. CHECK AGAINST TABLE III IN                5Ø6Ø
C PAGUROVA.                                                                5Ø7Ø
C                                                                          5Ø8Ø
      WRITE (6,99996)                                                      5Ø9Ø
      DO 9Ø I=1,6                                                          51ØØ
        X = X2(I)                                                          511Ø
        DX = DX2(I)                                                        512Ø
        DO 8Ø J=1,11                                                       513Ø
          ANU = FLOAT(J-1)*.1                                              514Ø
          A = -ANU + 1.                                                    515Ø
          DANU = DBLE(FLOAT(J-1))*.1DØ                                     516Ø
          DA = -DANU + 1.DØ                                                517Ø
          CALL GAM(A, X, 7., G, GSTAR, IFLG, IFLGST)                       518Ø
          CALL DGAM(DA, DX, 16., DG, DGSTAR, IFGD, IFGSTD)                 519Ø
          IF (J.LT.11) GO TO 6Ø                                            52ØØ
          ESUBNU = EXP(-X)*G                                               521Ø
          DESBNU = DEXP(-DX)*DG                                            522Ø
          GO TO 7Ø                                                         523Ø
   6Ø     ESUBNU = X**(-A)*EXP(ALGA(A+1.))*G/A                             524Ø
          DESBNU = DX**(-DA)*DEXP(DLGA(DA+1.DØ))*DG/DA                     525Ø
   7Ø     ERROR = SNGL(DABS(DBLE(ESUBNU)-DESBNU))                          526Ø
          IF (DESBNU.NE.Ø.DØ) ERROR = ABS(ERROR/SNGL(DESBNU))              527Ø
          WRITE (6,99995) ANU, A, X, ESUBNU, DESBNU, ERROR, IFLG,          528Ø
     *     IFLGST, IFGD, IFGSTD                                            529Ø
   8Ø   CONTINUE                                                           53ØØ
        WRITE (6,99998)                                                    531Ø
   9Ø CONTINUE                                                             532Ø
      STOP                                                                 533Ø
99999 FORMAT (/7X, 1HN, 5X, 1HA, 8X, 1HX, 8X, 8HESUBN(X), 1ØX, 6HESUBN(,  534Ø
     * 2HX), 1ØX, 5HERROR, 5X, 4HIFLG, 1X, 6HIFLGST, 2X, 4HIFGD, 1X,      535Ø
     * 6HIFGSTD/)                                                          536Ø
99998 FORMAT (6X, I2, E1Ø.1, E1Ø.2, E15.7, D22.15, E1Ø.2, 4I6)             537Ø
99997 FORMAT (/)                                                           538Ø
99996 FORMAT (//5X, 2HNU, 7X, 1HA, 8X, 1HX, 8X, 9HESUBNU(X), 9X,           539Ø
     * 9HESUBNU(X), 9X, 5HERROR, 5X, 4HIFLG, 1X, 6HIFLGST, 2X, 4HIFGD,    54ØØ
     * 1X, 6HIFGSTD/)                                                     541Ø
99995 FORMAT (1X, 2E9.1, E1Ø.2, E15.7, D22.15, E1Ø.2, 4I6)                 542Ø
      END                                                                  543Ø
C                                                                          544Ø
C DRIVER5 - CHISQUARE DISTRIBUTION                                         545Ø
C                                                                          546Ø
C P(CHISQUARE,NU) AND Q(CHISQUARE,NU) FOR SELECTED VALUES OF               547Ø
C CHISQUARE AND NU. CHECK AGAINST TABLE 26.7 IN NBS HANDBOOK.              548Ø
C                                                                          549Ø
      DIMENSION CCHSQ(9), NUMAX(9)                                         55ØØ
      DATA CCHSQ /.1,1.,2.,4.,6.,8.,15.,2Ø.,6Ø./                           551Ø
      DATA NUMAX /6,12,16,22,27,4*3Ø/                                      552Ø
      WRITE (6,99999)                                                      553Ø
      DO 2Ø I=1,9                                                          554Ø
        CHSQ = CCHSQ(I)                                                    555Ø
        NUI = NUMAX(I)                                                     556Ø
```

```
      DO 1Ø NU=1,NUI                                                    557Ø
        A = .5*FLOAT(NU)                                                558Ø
        X = .5*CHSQ                                                     559Ø
        CALL GAM(A, X, 5., Q, P, IFLG, IFLGST)                          56ØØ
        CALL GAM(A, X, 14., QØ, PØ, IFLGØ, IFLGSØ)                      561Ø
        ERRP = ABS(P-PØ)/PØ                                             562Ø
        ERRQ = ABS(Q-QØ)/QØ                                             563Ø
        WRITE (6,99998) NU, CHSQ, X, P, Q, ERRP, ERRQ, IFLG, IFLGST     564Ø
 1Ø   CONTINUE                                                          565Ø
      WRITE (6,99997)                                                   566Ø
 2Ø CONTINUE                                                            567Ø
    STOP                                                                568Ø
99999 FORMAT (//6X, 2HNU, 4X, 9HCHISQUARE, 7X, 1HX, 14X, 1HP, 14X, 1HQ,  569Ø
   * 11X, 7HERROR P, 8X, 7HERROR Q, 5X, 4HIFLG, 2X, 6HIFLGST/)          57ØØ
99998 FORMAT (5X, I3, 2E13.3, 4E15.4, 2I6)                              571Ø
99997 FORMAT (/)                                                        572Ø
      END                                                               573Ø


C                                                                       MAN7  1Ø
C DRIVER7 - MOLECULAR INTEGRALS                                         MAN7  2Ø
C ASUBN%ALPHAF FOR N#Ø%1F16 AND ALPHA#VAR TO 16 DECIMAL PLACES.         MAN7  3Ø
C CHECK AGAINST TABLES IN MILLER,GERHAUSEN AND MATSEN.                  MAN7  4Ø
C                                                                       MAN7  5Ø
      DOUBLE PRECISION ALPHA, X, P, A, G, GSTAR, DG, DGSTAR, ASUBN,     MAN7  6Ø
     * DASUBN                                                          MAN7  7Ø
      DIMENSION ALPHA%1ØF                                             MAN7  8Ø
      DATA ALPHA /.125DØ,.5DØ,1.625DØ,4.25DØ,7.375DØ,9.875DØ,12.625DØ, MAN7  9Ø
     * 17.125DØ,21.25DØ,25.DØ/                                         MAN7 1ØØ
      ACC # 16.                                                       MAN7 11Ø
      DACC # 25.                                                      MAN7 12Ø
      DO 4Ø I#1,1Ø                                                    MAN7 13Ø
        X # ALPHA%IF                                                  MAN7 14Ø
        WRITE %6,99999F X                                             MAN7 15Ø
        WRITE %6,99998F                                               MAN7 16Ø
        DO 3Ø J#1,17                                                  MAN7 17Ø
          N # J - 1                                                   MAN7 18Ø
          P # 1.DØ/X                                                  MAN7 19Ø
          IF %N.EQ.ØF GO TO 2Ø                                        MAN7 2ØØ
          DO 1Ø K#1,N                                                 MAN7 21Ø
            P # DBLE%FLOAT%KFF*P/X                                    MAN7 22Ø
 1Ø       CONTINUE                                                    MAN7 23Ø
 2Ø       A # DBLE%FLOAT%JFF                                          MAN7 24Ø
          CALL DGAM%A, X, ACC, G, GSTAR, IFLG, IFLGSTF                MAN7 25Ø
          CALL DGAM%A, X, DACC, DG, DGSTAR, IFLGD, IFLGSDF            MAN7 26Ø
          ASUBN # P*G                                                 MAN7 27Ø
          DASUBN # P*DG                                               MAN7 28Ø
          ERR # ABS%SNGL%%ASUBN-DASUBNF/DASUBNFF                      MAN7 29Ø
          WRITE %6,99997F N, ASUBN, DASUBN, ERR, IFLG, IFLGST, IFLGD, MAN7 3ØØ
   *       IFLGSD                                                     MAN7 31Ø
 3Ø     CONTINUE                                                      MAN7 32Ø
 4Ø CONTINUE                                                          MAN7 33Ø
    STOP                                                              MAN7 34Ø
99999 FORMAT %/5X, 6HALPHA#, D11.4//F                                 MAN7 35Ø
99998 FORMAT %12X, 1HN, 7X, 12HASUBN%ALPHAF, 17X, 12HASUBN%ALPHAF, 16X, MAN7 36Ø
   * 5HERROR, 6X, 4HIFLG, 1X, 6HIFLGST, 1X, 5HIFLGD, 1X, 6HIFLGSD/F   MAN7 37Ø
99997 FORMAT %1ØX, I3, D24.15, D33.24, E15.4, 4I6F                    MAN7 38Ø
      END                                                             MAN7 39Ø


      SUBROUTINE DGAM(A, X, ACC, G, GSTAR, IFLG, IFLGST)              574Ø
      DOUBLE PRECISION A, X, G, GSTAR, C, AL1Ø, ALX, ALPHA, ALPREC,   575Ø
     * TOP, BOT, AINF, EPS, EPS1, ES, SGA, AE, AA, AP1, AM1, AEP1,    576Ø
     * AEM1, FMA, AEPS, SGAE, AAEPS, ALGP1, SGGA, ALGEP1, SGGS, ALGS, 577Ø
     * ALG, SUM, GA, Y, TERM, U, P, Q, R, V, T, H, SGT, A1X, RHO, XPA, 578Ø
     * XMA, S, DLGA                                                   579Ø
      DIMENSION C(29)                                                 58ØØ
      DATA PREC, TOPEXP, BOTEXP /28.8989,322.,-293./                  581Ø
      DATA AL1Ø /2.3Ø2585Ø92994Ø4568 4Ø179914547DØ/                    582Ø
      DATA C /.5772156649Ø153286Ø6Ø6512Ø9ØØ8DØ,-.655878Ø7152Ø2538 81Ø77Ø1 583Ø
     * 951515DØ,-4.2ØØ2635Ø34Ø952355 29ØØ393488D-2,.166538611382291 4895Ø1 584Ø
     * 7ØØ7951DØ,-4.219773455554433 67482Ø83Ø13D-2,-9.6219715278769735621 585Ø
     * 149217D-3,7.218943246663Ø995 42395Ø1Ø3D-3,-1.165167591859Ø65112113 586Ø
     * 971D-3,-2.1524167411495Ø9728 1573ØD-4,1.28Ø5Ø2823881161861532ØD-4, 587Ø
     * -2.Ø1348547Ø78823865569D-5,-1.25Ø4934821426706573D-6,            588Ø
```

```
     * 1.133Ø272319816958824D-6,-2.Ø5633841697760710 3D-7,                    589Ø
     * 6.116Ø951Ø44814158D-9,5.ØØ2ØØ76444692229D-9,-1.1812745704870 2ØD-9     59ØØ
     * ,1.Ø4342671169110D-1Ø,7.782263439905D-12,-3.6968Ø5618642D-12,          591Ø
     * 5.1ØØ37Ø2875D-13,-2.Ø5832Ø654D-14,-5.34812254D-15,1.2267786D-15,       592Ø
     * -1.181259D-16,1.187D-18,1.412D-18,-2.3ØD-19,1.7D-2Ø/                   593Ø
      G = Ø.DØ                                                               594Ø
      GSTAR = Ø.DØ                                                           595Ø
      IF (X.LT.Ø.DØ) GO TO 29Ø                                               596Ø
C                                                                            597Ø
C INITIALIZATION                                                             598Ø
C                                                                            599Ø
      IFLG = Ø                                                               6ØØØ
      IFLGST = Ø                                                             6Ø1Ø
      I = Ø                                                                  6Ø2Ø
      IF (X.GT.Ø.DØ) ALX = DLOG(X)                                           6Ø3Ø
      ALPHA = X + .25DØ                                                      6Ø4Ø
      IF (X.LT..25DØ .AND. X.GT.Ø.DØ) ALPHA = DLOG(.5DØ)/ALX                 6Ø5Ø
      ALPREC = AL1Ø*DBLE(PREC)                                               6Ø6Ø
      TOP = AL1Ø*DBLE(TOPEXP)                                                6Ø7Ø
      BOT = AL1Ø*DBLE(BOTEXP)                                                6Ø8Ø
      AINF = 1Ø.DØ**TOPEXP                                                   6Ø9Ø
      EPS = .5DØ*1Ø.DØ**(-ACC)                                               61ØØ
      EPS1 = EPS/1.D2                                                        611Ø
      SGA = 1.DØ                                                             612Ø
      IF (A.LT.Ø.DØ) SGA = -SGA                                              613Ø
      AE = A                                                                 614Ø
      AA = DABS(A)                                                           615Ø
      AP1 = A + 1.DØ                                                         616Ø
      AEP1 = AP1                                                             617Ø
      MA = SNGL(.5DØ-A)                                                      618Ø
      FMA = DBLE(FLOAT(MA))                                                  619Ø
      AEPS = A + FMA                                                         62ØØ
      SGAE = 1.DØ                                                            621Ø
      IF (AEPS.LT.Ø.DØ) SGAE = -SGAE                                         622Ø
      AAEPS = DABS(AEPS)                                                     623Ø
      ALGP1 = Ø.DØ                                                           624Ø
C                                                                            625Ø
C EVALUATION OF THE LOGARITHM OF THE ABSOLUTE VALUE OF                       626Ø
C GAMMA(A+1.) AND DETERMINATION OF THE SIGN OF GAMMA(A+1.)                   627Ø
C                                                                            628Ø
      SGGA = 1.DØ                                                            629Ø
      IF (MA.LE.Ø) GO TO 1Ø                                                  63ØØ
      IF (AEPS.EQ.Ø.DØ) GO TO 2Ø                                             631Ø
      SGGA = SGAE                                                            632Ø
      IF (MA.EQ.2*(MA/2)) SGGA = -SGGA                                       633Ø
      ALGP1 = DLGA(AEPS+1.DØ) - DLOG(AAEPS)                                  634Ø
      IF (MA.EQ.1) GO TO 2Ø                                                  635Ø
      ALGP1 = ALGP1 + DLGA(1.DØ-AEPS) - DLGA(FMA-AEPS)                       636Ø
      GO TO 2Ø                                                               637Ø
   1Ø ALGP1 = DLGA(AP1)                                                      638Ø
   2Ø ALGEP1 = ALGP1                                                         639Ø
      IF (X.GT.Ø.DØ) GO TO 6Ø                                                64ØØ
C                                                                            641Ø
C EVALUATION OF GSTAR(A,Ø.) AND G(A,Ø.)                                      642Ø
C                                                                            643Ø
      IF (A) 3Ø, 4Ø, 5Ø                                                      644Ø
   3Ø IFLGST = 2                                                             645Ø
      GSTAR = AINF                                                           646Ø
      G = 1.DØ/AA                                                            647Ø
      RETURN                                                                 648Ø
   4Ø IFLGST = 3                                                             649Ø
      GSTAR = 1.DØ                                                           65ØØ
      IFLG = 2                                                               651Ø
      G = AINF                                                               652Ø
      RETURN                                                                 653Ø
   5Ø G = 1.DØ                                                               654Ø
      RETURN                                                                 655Ø
   6Ø IF (A.GT.ALPHA) GO TO 22Ø                                              656Ø
      IF (X.GT.1.5DØ) GO TO 24Ø                                              657Ø
      IF (A.LT.-.5DØ) GO TO 17Ø                                              658Ø
C                                                                            659Ø
C DIRECT EVALUATION OF G(A,X) AND GSTAR(A,X) FOR X.LE.1.5                    66ØØ
C AND -.5.LE.A.LE.ALPHA(X)                                                   661Ø
C                                                                            662Ø
      GSTAR = 1.DØ                                                           663Ø
      IF (A.GE..5DØ) GO TO 11Ø                                               664Ø
```

```
    7Ø SUM = C(29)                                                    665Ø
       DO 8Ø K=1,28                                                   666Ø
         K1 = 29 - K                                                  667Ø
         SUM = AE*SUM + C(K1)                                         668Ø
    8Ø CONTINUE                                                       669Ø
       GA = -SUM/(1.DØ+AE*SUM)                                        67ØØ
       Y = AE*ALX                                                     671Ø
       IF (DABS(Y).GE.1.DØ) GO TO 1ØØ                                 672Ø
       SUM = 1.DØ                                                     673Ø
       TERM = 1.DØ                                                    674Ø
       K = 1                                                          675Ø
    9Ø K = K + 1                                                      676Ø
       IF (K.GT.6ØØ) GO TO 33Ø                                        677Ø
       TERM = Y*TERM/DBLE(FLOAT(K))                                   678Ø
       SUM = SUM + TERM                                               679Ø
       IF (DABS(TERM).GT.EPS1*SUM) GO TO 9Ø                           68ØØ
       U = GA - SUM*ALX                                               681Ø
       GO TO 12Ø                                                      682Ø
   1ØØ U = GA - (DEXP(Y)-1.DØ)/AE                                     683Ø
       GO TO 12Ø                                                      684Ø
   11Ø U = DEXP(DLGA(A)) - (X**A)/A                                   685Ø
   12Ø P = AE*X                                                       686Ø
       Q = AEP1                                                       687Ø
       R = AE + 3.DØ                                                  688Ø
       TERM = 1.DØ                                                    689Ø
       SUM = 1.DØ                                                     69ØØ
       K = 1                                                          691Ø
   13Ø K = K + 1                                                      692Ø
       IF (K.GT.6ØØ) GO TO 33Ø                                        693Ø
       P = P + X                                                      694Ø
       Q = Q + R                                                      695Ø
       R = R + 2.DØ                                                   696Ø
       TERM = -P*TERM/Q                                               697Ø
       SUM = SUM + TERM                                               698Ø
       IF (DABS(TERM).GT.EPS1*SUM) GO TO 13Ø                          699Ø
       V = (X**AEP1)*SUM/AEP1                                         7ØØØ
       G = U + V                                                      7Ø1Ø
       IF (I.EQ.1) GO TO 18Ø                                          7Ø2Ø
       IF (A) 14Ø, 15Ø, 16Ø                                           7Ø3Ø
   14Ø T = DEXP(X)*X**(-A)                                            7Ø4Ø
       G = T*G                                                        7Ø5Ø
       GSTAR = 1.DØ - A*G*DEXP(-ALGP1)/T                              7Ø6Ø
       RETURN                                                         7Ø7Ø
   15Ø G = DEXP(X)*G                                                  7Ø8Ø
       RETURN                                                         7Ø9Ø
   16Ø G = A*G*DEXP(-ALGP1)                                           71ØØ
       GSTAR = 1.DØ - G                                               711Ø
       RETURN                                                         712Ø
C                                                                     713Ø
C RECURSIVE EVALUATION OF G(A,X) FOR X.LE.1.5 AND A.LT.-.5            714Ø
C                                                                     715Ø
   17Ø I = 1                                                          716Ø
       AE = AEPS                                                      717Ø
       AEP1 = AEPS + 1.DØ                                             718Ø
       IF (X.LT..25DØ .AND. AE.GT.ALPHA) GO TO 21Ø                    719Ø
       GO TO 7Ø                                                       72ØØ
   18Ø G = G*DEXP(X)*X**(-AE)                                         721Ø
       DO 19Ø K=1,MA                                                  722Ø
         G = (1.DØ-X*G)/(DBLE(FLOAT(K))-AE)                           723Ø
   19Ø CONTINUE                                                       724Ø
       ALG = DLOG(G)                                                  725Ø
C                                                                     726Ø
C EVALUATION OF GSTAR(A,X) IN TERMS OF G(A,X)                         727Ø
C                                                                     728Ø
   2ØØ GSTAR = 1.DØ                                                   729Ø
       IF (MA.GE.Ø .AND. AEPS.EQ.Ø.DØ) RETURN                         73ØØ
       SGT = SGA*SGGA                                                 731Ø
       T = DLOG(AA) - X + A*ALX + ALG - ALGP1                         732Ø
       IF (T.LT.-ALPREC) RETURN                                       733Ø
       IF (T.GE.TOP) GO TO 32Ø                                        734Ø
       GSTAR = 1.DØ - SGT*DEXP(T)                                     735Ø
       RETURN                                                         736Ø
   21Ø I = 2                                                          737Ø
       ALGEP1 = DLGA(AEP1)                                            738Ø
C                                                                     739Ø
C EVALUATION OF GSTAR(A,X) FOR A.GT.ALPHA(X) BY TAYLOR                74ØØ
```

```
C EXPANSION                                                          741Ø
C                                                                    742Ø
  22Ø G = 1.DØ                                                       743Ø
      TERM = 1.DØ                                                    744Ø
      SUM = 1.DØ                                                     745Ø
      K = Ø                                                          746Ø
  23Ø K = K + 1                                                      747Ø
      IF (K.GT.6ØØ) GO TO 34Ø                                        748Ø
      TERM = X*TERM/(AE+DBLE(FLOAT(K)))                              749Ø
      SUM = SUM + TERM                                               75ØØ
      IF (DABS(TERM).GT.EPS*SUM) GO TO 23Ø                           751Ø
      ALGS = AE*ALX - X + DLOG(SUM) - ALGEP1                         752Ø
      IF (ALGS.LE.BOT) GO TO 31Ø                                     753Ø
      GSTAR = DEXP(ALGS)                                             754Ø
      G = 1.DØ - GSTAR                                               755Ø
      IF (I.NE.2) RETURN                                             756Ø
      G = G*DEXP(ALGEP1)/AE                                          757Ø
      GO TO 18Ø                                                      758Ø
C                                                                    759Ø
C EVALUATION OF G(A,X) FOR X.GT.1.5 AND A.LE.ALPHA(X) BY             76ØØ
C MEANS OF THE LEGENDRE CONTINUED FRACTION                           761Ø
C                                                                    762Ø
  24Ø GSTAR = 1.DØ                                                   763Ø
      XPA = X + 1.DØ - A                                             764Ø
      XMA = X - 1.DØ - A                                             765Ø
      P = Ø.DØ                                                       766Ø
      Q = XPA*XMA                                                    767Ø
      R = 4.DØ*XPA                                                   768Ø
      S = -A + 1.DØ                                                  769Ø
      TERM = 1.DØ                                                    77ØØ
      SUM = 1.DØ                                                     771Ø
      RHO = Ø.DØ                                                     772Ø
      K = 1                                                          773Ø
  25Ø K = K + 1                                                      774Ø
      IF (K.GT.6ØØ) GO TO 33Ø                                        775Ø
      P = P + S                                                      776Ø
      Q = Q + R                                                      777Ø
      R = R + 8.DØ                                                   778Ø
      S = S + 2.DØ                                                   779Ø
      RHO = T/(Q-T)                                                  781Ø
      TERM = RHO*TERM                                                782Ø
      SUM = SUM + TERM                                               783Ø
      IF (DABS(TERM).GT.EPS*SUM) GO TO 25Ø                           784Ø
      IF (A) 26Ø, 27Ø, 28Ø                                           785Ø
  26Ø G = SUM/XPA                                                    786Ø
      ALG = DLOG(G)                                                  787Ø
      GO TO 2ØØ                                                      788Ø
  27Ø G = SUM/XPA                                                    789Ø
      RETURN                                                         79ØØ
  28Ø ALG = A*ALX - X + DLOG(A*SUM/XPA) - ALGP1                      791Ø
      IF (ALG.LE.BOT) GO TO 3ØØ                                      792Ø
      G = DEXP(ALG)                                                  793Ø
      GSTAR = 1.DØ - G                                               794Ø
      RETURN                                                         795Ø
  29Ø IFLG = 1                                                       796Ø
      IFLGST = 1                                                     797Ø
      RETURN                                                         798Ø
  3ØØ IFLG = 4                                                       799Ø
      RETURN                                                         8ØØØ
  31Ø IFLGST = 4                                                     8Ø1Ø
      RETURN                                                         8Ø2Ø
  32Ø IFLGST = 5                                                     8Ø3Ø
      GSTAR = -SGT*AINF                                              8Ø4Ø
      RETURN                                                         8Ø5Ø
  33Ø IFLG = 6                                                       8Ø6Ø
      RETURN                                                         8Ø7Ø
  34Ø IFLGST = 6                                                     8Ø8Ø
      RETURN                                                         8Ø9Ø
      END                                                            81ØØ
      DOUBLE PRECISION FUNCTION DLGA(DX)                             811Ø
      DOUBLE PRECISION DBNUM, DBDEN, DX, DC, DP, DY, DT, DS          812Ø
      DIMENSION DBNUM(8), DBDEN(8)                                   813Ø
      DATA DBNUM /-3.617D3,1.DØ,-6.91D2,1.DØ,-1.DØ,1.DØ,-1.DØ,1.DØ/, 814Ø
     * DBDEN /1.224D5,1.56D2,3.6Ø36D5,1.188D3,1.68D3,1.26D3,3.6D2,1.2D1/ 815Ø
      DC = .5DØ*DLOG(8.DØ*DATAN(1.DØ))                               816Ø
      DP = 1.DØ                                                      817Ø
```

```
      DY = DX                                                          818Ø
      Y = SNGL(DY)                                                     819Ø
C                                                                      82ØØ
C THE CONDITIONAL CLAUSE IN THE NEXT STATEMENT EXPRESSES THE           821Ø
C INEQUALITY Y.GT.EXP(.121189*DPREC+.Ø539Ø5), WHERE DPREC IS THE       822Ø
C NUMBER OF DECIMAL DIGITS CARRIED IN DOUBLE PRECISION FLOATING-POINT  823Ø
C ARITHMETIC.                                                          824Ø
C                                                                      825Ø
   1Ø IF (Y.GT.35.Ø27) GO TO 2Ø                                        826Ø
      DP = DY*DP                                                       827Ø
      DY = DY + 1.DØ                                                   828Ø
      Y = SNGL(DY)                                                     829Ø
      GO TO 1Ø                                                         83ØØ
   2Ø DT = 1.DØ/(DY*DY)                                                831Ø
      DS = 4.3867D4/2.44188D5                                          832Ø
      DO 3Ø I=1,8                                                      833Ø
        DS = DT*DS + DBNUM(I)/DBDEN(I)                                 834Ø
   3Ø CONTINUE                                                         835Ø
      DLGA = (DY-.5DØ)*DLOG(DY) - DY + DC + DS/DY - DLOG(DP)           836Ø
      RETURN                                                           837Ø
      END                                                              838Ø
```

ALGORITHM 543
FFT9, Fast Solution of Helmholtz-Type Partial Differential Equations [D3]

E. N. HOUSTIS
Purdue University
and
T. S. PAPATHEODOROU
Clarkson College of Technology

Key Words and Phrases: fast Fourier transform, fast Helmholtz solver, fast Poisson solver
CR Categories: 5.17
Language: Fortran

DESCRIPTION

The algorithm given here is a complement to [1] where the description, test results, and references are given.

REFERENCES

1. HOUSTIS, E.N., AND PAPATHEODOROU, T.S. High-order fast elliptic equation solver. *ACM Trans. Math. Software 5*, 4 (Dec. 1979), 431-441.

ALGORITHM

```
C     PROGRAM FFT9(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)                  1Ø
C                                                                          2Ø
C         ---- PROGRAM DESCRIPTION ----                                    3Ø
C     PROGRAM FFT9 USES A 4-TH OR 6-TH ORDER 9-POINT DIFFERENCE            4Ø
C     FORMULA AND FAST FOURIER TRANSFORM FOR THE NUMERICAL                 5Ø
C     SOLUTION OF THE ELLIPTIC EQUATION WITH CONSTANT COEFFICIENTS         6Ø
C                                                                          7Ø
C     (I)  CUXX*DDXU + CUYY*DDYU + CU*U = R                                8Ø
C                                                                          9Ø
C     ON A RECTANGULAR REGION Ø .LE. X .LE. SX,Ø .LE. Y .LE. SY           1ØØ
C     AND SUBJECT TO DIRICHLET BOUNDARY CONDITIONS                        11Ø
C     (II)  U = G                ON THE BOUNDARY                          12Ø
C                                                                         13Ø
C     NOTE- THE 6-TH ORDER ALGORITHM IS APPLIED ONLY                      14Ø
C           TO POISSON TYPE OPERATORS                                     15Ø
C                                                                         16Ø
C                                                                         17Ø
C         ---- INPUT AND OUTPUT TO FFT9 ----                             18Ø
C     --PROBLEM DEFINITION-- USER SUPPLIED FORTRAN FUNCTION FOR THE       19Ø
C     EVALUATION OF THE RIGHT SIDES (R,G) OF THE DIFFERENTIAL             2ØØ
```

Received 31 May 1977; revised 2 January 1979.

This work was supported in part by the National Science Foundation under Grant MCS 76-10225.
Authors' addresses: E. N. Houstis, Department of Computer Science, Purdue University, West Lafayette, IN 47907; T. S. Papatheodorou, Department of Mathematics, Clarkson College of Technology, Potsdam, NY 13676.

```
C     AND BOUNDARY OPERATORS                                          21Ø
C                                                                     22Ø
C     REAL       FUNCTION PDERGH(X,Y)                               23Ø
C                                                                     24Ø
C     PDERGH = R                                                      25Ø
C     RETURN                                                          26Ø
C     END                                                             27Ø
C     REAL       FUNCTION BCOND(I,X,Y,BVALUS)                       28Ø
C        TWO DIMENSIONS                                               29Ø
C        VALUES OF BOUNDARY CONDITION  ON SIDE I                      3ØØ
C        AT (X,Y)                                                     31Ø
C                                                                     32Ø
C             I=4                                                     33Ø
C          ---------------                                            34Ø
C          I             I                                           35Ø
C          I             I                                           36Ø
C     I=3  I   REGION     I  I=1                                      37Ø
C          I             I                                           38Ø
C          I             I                                           39Ø
C          ---------------                                            4ØØ
C             I=2                                                     41Ø
C                                                                     42Ø
C          REAL           BVALUS(4)                                   43Ø
C          GO TO(1ØØ,1Ø1,1Ø2,1Ø3) , I                                 44Ø
C     1Ø1   BVALUS(4) = G                                             45Ø
C          BCOND = BVALUS(4)                                          46Ø
C          RETURN                                                     47Ø
C     1Ø2   BVALUS(4) = G                                             48Ø
C          BCOND = BVALUS(4)                                          49Ø
C          RETURN                                                     5ØØ
C     1Ø3   BVALUS(4) = G                                             51Ø
C          BCOND = BVALUS(4)                                          52Ø
C          RETURN                                                     53Ø
C     1Ø4   BVALUS(4) = G                                             54Ø
C          BCOND = BVALUS(4)                                          55Ø
C          RETURN                                                     56Ø
C          END                                                        57Ø
C                                                                     58Ø
C     USER SUPPLIED SUBROUTINE FOR THE DEFINITION OF                  59Ø
C     P.D.E CONSTANT COEFFICIENTS                                     6ØØ
C                                                                     61Ø
C     SUBROUTINE PDE(X,Y,CVALUS)                                      62Ø
C                                                                     63Ø
C     REAL CVALUS(7)                                                  64Ø
C     CVALUS(1) = CUXX                                                65Ø
C     CVALUS(3) = CUYY                                                66Ø
C     CVALUS(6) = CU                                                  67Ø
C     RETURN                                                          68Ø
C     END                                                             69Ø
C                                                                     7ØØ
C                                                                     71Ø
C     USER SUPPLIED FORTRAN FUNCTION FOR THE TRUE SOLUTION            72Ø
C     IF KNOWN                                                        73Ø
C                                                                     74Ø
C          REAL       FUNCTION TRUE(X,Y)                            75Ø
C          TRUE = ...                                                 76Ø
C          RETURN                                                     77Ø
C          END                                                        78Ø
C                                                                     79Ø
C     --REGION AND GRID SPECIFICATIONS--                              8ØØ
C                                                                     81Ø
C          SX,SY          - LENGTHS OF SIDES OF RECTANGLE             82Ø
C          NGRIDX,NGRIDY  - NUMBER OF HORIZONTAL AND                  83Ø
C                           VERTICAL MESH LINES                       84Ø
C          IQX,IQY        - EXPONENTS OF 2                            85Ø
C                           NGRIDX=2**IQX+1,NGRIDY=2**IQY+1           86Ø
C                                                                     87Ø
C          READ(5,1ØØ) SX,SY,NGRIDX,NGRIDY                           88Ø
C     1ØØ  FORMAT(2F1Ø.Ø,2I3)                                         89Ø
C                                                                     9ØØ
C     --OUTPUT CONTROL--USER SUPPLIED DATA                            91Ø
C          LEVEL          - OUTPUT LEVEL DESIRED                     92Ø
C          NRUNS          - NUMBER OF SUCCESIVE RUNS.                93Ø
C                           IN EACH RUN THE MESH SIZE IS              94Ø
C                           CUT BY A FACTOR OF 2.                     95Ø
C          ORDER          - RATE OF CONVERGENCE DESIRED              96Ø
```

```
C         READ(5,102) LEVEL,NRUNS,ORDER                                  970
C     102  FORMAT(3I2)                                                   980
C                                                                        990
C         IF LEVEL = 0 PRINT APPROXIMATE SOLUTION AT NODES              1000
C                  = 1 PRINT MAXIMUM ERROR AND MAXIMUM                  1010
C                      RELATIVE ERROR PROVIDED TRUE SOLUTION            1020
C                      IS KNOWN                                         1030
C                   = 2 ALSO PRINT TRUE SOLUTION AND                    1040
C                       APPROXIMATE SOLUTION AT THE INTERIOR            1050
C                       GRID POINTS                                     1060
C                                                                       1070
C         IF ORDER = 4 CHOOSE A 4-TH ORDER DIFFERENCE APPR. TO (I)      1080
C                  = 6 CHOOSE A 6-TH ORDER DIFFERENCE APPR.TO (I)       1090
C     --STORAGE--IT IS ASSUMED THAT 3 .LE. IXQ,IQY .LE. 7 .             1100
C     IN CASE OF FINER MESH THE DIMENSIONS REQUIRED ARE                 1110
C     COMPUTED BY FORMULAS GIVEN IN ARRAY LIST DESCRIPTION.             1120
C        ---- MAIN   VARIABLES OF FFT9 ----                            1130
C                                                                       1140
C        WORK     - WORKING SPACE OF DIMENSION                         1150
C     ORDER = 6          7+10*MAX(NGRIDX,NGRIDY)+2*NGRIDX*NGRIDY       1160
C        Z,Y      - ARRAYS USED IN THE FOURIER ANALYSIS-SYNTHESIS      1170
C     ORDER = 4          7+10*MAX(NGRIDX,NGRIDY)+NGRIDX*NGRIDY         1180
C                  DIMENSION OF 'Z,Y' IS NZD=NYD= MAX(NX,NY)           1190
C        AKX(I+1)-THE I-TH EIGENVALUE OF THE DIAGONAL BLOCK DIVIDED BY 1200
C                 THE I-TH EIGENVALUE OF THE OFF DIAGONAL BLOCK         1210
C                 SQUARED MINUS 2                                       1220
C                 DIMENSION OF 'AKX' IS NAKXD = MAX(NX,NY)+2            1230
C                                                                       1240
C                                                                       1250
C        CORE(K) - VALUE OF APPROXIMATE SOLUTION AT NODE K              1260
C                  DIMENSION OF 'CORE' IS NCORED = NX+2+(NX+1)*(NY+1)   1270
C        PTINT( )- VALUES OF  G  AT HALF LATTICE POINTS                 1280
C                  DIMENSION IS NPINTD = NX*NY                          1290
C       NOTE- ARRAY 'PTINT' IS NOT USED IN CASE ORDER .EQ. 4            1300
C                                                                       1310
C        GRIDX,GRIDY     - GRID COORDINATES                             1320
C                          DIMENSION OF 'GRIDX,GRIDY' IS                1330
C                          NGRDXD = NX+1,NGRDYD = NY+1                  1340
C                                                                       1350
C        SX,SY           - LENGTHS OF RECTANGULAR REGION                1360
C                                                                       1370
C        ED(I+1)         - THE I-TH EIGENVALUE OF THE OFF DIAGONAL      1380
C                          BLOCK.DIMENSION OF 'ED' IS NEDD = MAX(NX,NY)+2 1390
C        INDEX           - INDEX VECTOR USED IN FFT ANALYSIS            1400
C                          DIMENSION IS INDEXD = MAX(NX,NY)             1410
C        SI              - SINES VECTOR FOR FFT ANALYSIS AND            1420
C                          SYNTHESIS. DIMENSION NSID = MAX(NX,NY)       1430
C                                                                       1440
C     ---- COMMON VARIABLES OF FFT9 ----                                1450
C                                                                       1460
C     COMMON /MESH/                                                     1470
C        NX,NY           - NX = 2**IQX,NY = 2**IQY                      1480
C        IMIN,IMAX       - RANGE OF INTERIOR NODES IN X-DIRECTION       1490
C        JMIN,JMAX       - RANGE OF INTERIOR NODES IN Y-DIRECTION       1500
C        INC             - INC = IMAX - IMIN + 3                        1510
C        IRO             - IRO = NX + 3                                 1520
C        IBCX            - IBCX = 1                                     1530
C        IQX             - EXPONENT OF 2                                1540
C        IBCY            - IBCY = 1                                     1550
C        IQY             - EXPONENT OF 2                                1560
C        HX,HY           - MESH SIZE                                    1570
C        HXY2            - HXY2 = (HX/HY)**2                            1580
C        PI              - PI = 3.14...                                 1590
C        POTFAC          - NORMALIZATION FACTOR = 2/NX                  1600
C     COMMON /FDFORM/                                                   1610
C        DLEFT           - DIAGONAL ENTRY OF THE OFFDIAGONAL BLOCK      1620
C        DIAG,OFFD       - DIAGONAL AND OFFDIAGONAL ENTRIES OF THE      1630
C                          DIAGONAL BLOCK OF THE 9-POINT FORMULA        1640
C        RF,HH,CH,RC1,FACTOR - CONSTANTS                                1650
C     COMMON /FFT/                                                      1660
C        ...             - LOCAL VARIABLE                               1670
C        ---- COMMON VARIABLES DECLARATION ----                         1680
C                                                                       1690
      COMMON /MESH/ NX,NY,IMIN,IMAX,JMIN,JMAX,INC,IRO,IBCX,IQX,IBCY,IQY, 1700
     1HX,HY,HXY2,PI,POTFAC,SX,SY                                        1710
      COMMON /FDFORM/ DLEFT,DIAG,OFFD,RF,HH,CH,RC1,FACTOR               1720
```

```
      COMMON /CPDE/ CUXX,CUYY,CU                                        173Ø
      COMMON /FFT/ N2,N4,N3,N7,IP,ISL,L1,IBCJ                           174Ø
      COMMON WORK(35Ø4Ø)                                                175Ø
C                                                                       176Ø
      REAL CVALUS(7)                                                    177Ø
      INTEGER ORDER                                                     178Ø
C                                                                       179Ø
C     INITIALIZATIONS                                                   18ØØ
C                                                                       181Ø
      IBCX=1                                                            182Ø
      IBCY=1                                                            183Ø
C                                                                       184Ø
C ***** INPUT *****                                                     185Ø
C     DEFINE P.D.E COEFFICIENTS                                         186Ø
C                                                                       187Ø
      CALL PDE (X,Y,CVALUS)                                             188Ø
C                                                                       189Ø
      CUXX=CVALUS(1)                                                    19ØØ
      CUYY=CVALUS(3)                                                    191Ø
      CU=CVALUS(6)                                                      192Ø
C                                                                       193Ø
C     DEFINE GRID SPECIFICATIONS                                        194Ø
C                                                                       195Ø
      READ (5,1Ø2) SX,SY,NGRIDX,NGRIDY                                  196Ø
      NX=NGRIDX-1                                                       197Ø
      NY=NGRIDY-1                                                       198Ø
      RNX=NX                                                            199Ø
      RNY=NY                                                            2ØØØ
      RALOG2=1./ALOG(2.)                                                2Ø1Ø
      IQX=ALOG(RNX)*RALOG2                                              2Ø2Ø
      IQY=ALOG(RNY)*RALOG2                                              2Ø3Ø
C                                                                       2Ø4Ø
C     DEFINE OUTPUT CONTROL AND ORDER OF FINITE DIFFERENCE              2Ø5Ø
C     DESCRITIZATION FORMULA                                            2Ø6Ø
C                                                                       2Ø7Ø
      READ (5,1Ø3) LEVEL,NRUNS,ORDER                                    2Ø8Ø
C                                                                       2Ø9Ø
C     OUTPUT THE INPUT DATA                                             21ØØ
C                                                                       211Ø
      WRITE (6,1Ø4)                                                     212Ø
      WRITE (6,1Ø5)                                                     213Ø
      WRITE (6,1Ø6) CUXX,CUYY,CU                                        214Ø
      WRITE (6,1Ø7)                                                     215Ø
      WRITE (6,1Ø8) SX,SY                                               216Ø
      WRITE (6,1Ø9)                                                     217Ø
      WRITE (6,11Ø) ORDER                                               218Ø
      DO 1Ø1 NTIMES=1,NRUNS                                             219Ø
C                                                                       22ØØ
C ***** DISCRETIZATION *****                                            221Ø
C     APPROXIMATE THE DIFFERENCIAL EQUATION WITH 9-POINT DIFF.OPER.     222Ø
C                                                                       223Ø
         NAKXD=MAXØ(NX,NY)+2                                            224Ø
         NEDD=NAKXD                                                     225Ø
         NGRDXD=NX+1                                                    226Ø
         NGRDYD=NY+1                                                    227Ø
C                                                                       228Ø
         IA1=1                                                          229Ø
         IA2=IA1+NAKXD                                                  23ØØ
         IA3=IA2+NEDD                                                   231Ø
         IA4=IA3+NGRDXD                                                 232Ø
         IA5=IA4+NGRDYD                                                 233Ø
         CALL DISCRT (ORDER,WORK(IA1),NAKXD,WORK(IA2),NEDD,WORK(IA3),NGR 234Ø
     1   DXD,WORK(IA4),NGRDYD)                                          235Ø
C                                                                       236Ø
C     GENERATE RIGHT SIDE OF DIFFERENCE EQUATIONS                       237Ø
C                                                                       238Ø
         NCORED=NGRDXD*NGRDYD+NAKXD                                     239Ø
         NPINTD=NCORED                                                  24ØØ
         IF (ORDER.EQ.4) NPINTD=1                                       241Ø
         IA6=IA5+NCORED                                                 242Ø
         IA7=IA6+NPINTD                                                 243Ø
C                                                                       244Ø
         CALL RGHTSD (ORDER,WORK(IA5),NCORED,WORK(IA6),NPINTD,WORK(IA3), 245Ø
     1   NGRDXD,WORK(IA4),NGRDYD)                                       246Ø
C                                                                       247Ø
C ***** EQUATION SOLUTION ******                                        248Ø
```

```
C                                                                       249Ø
C     GENERATE INDECIES AND SINES USED IN THE FOURIER ANALYSIS          25ØØ
C     AND SYNTHESIS                                                     251Ø
C                                                                       252Ø
         INDEXD=MAXØ(NX,NY)                                             253Ø
         NSID=INDEXD                                                    254Ø
         IA8=IA7+INDEXD                                                 255Ø
         IA9=IA8+NSID                                                   256Ø
         CALL SETF (IBCX,IQX,WORK(IA7),INDEXD,WORK(IA8),NSID)           257Ø
C                                                                       258Ø
C                                                                       259Ø
C     SOLVE THE BLOCK TRIDIAGONAL SYSTEM OF DIFFERENCE EQUATIONS        26ØØ
C     WITH THE FAST FOURIER SERIES METHOD.                              261Ø
C                                                                       262Ø
         NZD=NAKXD                                                      263Ø
         IA1Ø=IA9+NZD                                                   264Ø
         NYD=NAKXD                                                      265Ø
         CALL EQSOL (WORK(IA5),NCORED,WORK(IA9),NZD,WORK(IA8),NSID,WORK(  266Ø
     1   IA1Ø),NYD,WORK(IA7),INDEXD,WORK(IA2),NEDD,WORK(IA1),NAKXD)      267Ø
C                                                                       268Ø
C ***** OUTPUT ******                                                   269Ø
C                                                                       27ØØ
C                                                                       271Ø
C     PRINTS THE COMPUTED SOLUTION AND MAX.ERROR,MAX.RELATIVE           272Ø
C     ERROR IF THE SOLUTION IS KNOWN                                    273Ø
C                                                                       274Ø
         CALL SUMARY (LEVEL,WORK(IA3),NGRDXD,WORK(IA4),NGRDYD,WORK(IA5),  275Ø
     1   NCORED)                                                        276Ø
C                                                                       277Ø
C     INCREASE EXPONENT OF 2                                            278Ø
C                                                                       279Ø
         IQX=IQX+1                                                      28ØØ
         IQY=IQY+1                                                      281Ø
         NX=2**IQX                                                      282Ø
         NY=2**IQY                                                      283Ø
  1Ø1 CONTINUE                                                          284Ø
      STOP                                                              285Ø
C                                                                       286Ø
  1Ø2 FORMAT (2F1Ø.Ø,2I3)                                               287Ø
  1Ø3 FORMAT (3I2)                                                      288Ø
  1Ø4 FORMAT (1X,27H******* INPUT DATA ********)                        289Ø
  1Ø5 FORMAT (1X,37HEQUATION.  CUXX*DDXU+CUYY*DDYU+CU*U=R)              29ØØ
  1Ø6 FORMAT (1X,19HCOEFFICIENTS. CUXX=,F1Ø.3,5HCUYY=,F1Ø.3,3HCU=,F1Ø.3)  291Ø
  1Ø7 FORMAT (1X,2ØHBOUNDARY COND. U = G)                               292Ø
  1Ø8 FORMAT (1X,24HREGION.  Ø. .LE. X .LE. ,F8.3,3X,15HØ. .LE. Y .LE. ,  293Ø
     1F8.3)                                                             294Ø
  1Ø9 FORMAT (1X,25H******* SOLUTION ********)                          295Ø
  11Ø FORMAT (1X,15HDISCRETIZATION.,5X,I5,8H- ORDER ,24HDIFFERENCE APPRO  296Ø
     1XIMATION)                                                         297Ø
C                                                                       298Ø
      END                                                               299Ø
      SUBROUTINE RGHTSD (ORDER,CORE,NCORED,PTINT,NPINTD,GRIDX,NGRDXD,GRI  3ØØØ
     1DY,NGRDYD)                                                        3Ø1Ø
C                                                                       3Ø2Ø
C     COMPUTES RIGHT SIDE OF THE FINITE DIFFERENCE EQUATIONS            3Ø3Ø
C                                                                       3Ø4Ø
C     THE ARGUMENTS - ORDER,CORE(NCORED),PTINT(NPINTD),GRIDX(NGRDXD),   3Ø5Ø
C                     GRIDY(NGRDYD) - DEFINED IN FFT9 MAIN PROGRAM      3Ø6Ø
C                                                                       3Ø7Ø
      INTEGER ORDER                                                     3Ø8Ø
      REAL CORE(NCORED),PTINT(NPINTD),GRIDX(NGRDXD),GRIDY(NGRDYD),BVALUS  3Ø9Ø
     1(4)                                                               31ØØ
C                                                                       311Ø
C     FFT9 COMMON VARIABLES                                             312Ø
C                                                                       313Ø
      COMMON /MESH/ NX,NY,IMIN,IMAX,JMIN,JMAX,INC,IRO,IBCX,IQX,IBCY,IQY,  314Ø
     1HX,HY,HXY2,PI,POTFAC,SX,SY                                        315Ø
      COMMON /FDFORM/ DLEFT,DIAG,OFFD,RF,HH,CH,RC1,FACTOR               316Ø
C                                                                       317Ø
C     INITIALIZATIONS                                                   318Ø
C                                                                       319Ø
      IØ=Ø                                                              32ØØ
      JØ=Ø                                                              321Ø
      ZØ=Ø.Ø                                                            322Ø
      Z1=1.Ø                                                            323Ø
C                                                                       324Ø
```

```
C     FUNCTION EVALUATIONS                                                    325Ø
C                                                                             326Ø
      DO 1Ø2 J=JØ,NY                                                          327Ø
         W=GRIDY(J+1)                                                         328Ø
         L=IRO+INC*J                                                          329Ø
         DO 1Ø1 I=IØ,NX                                                       33ØØ
            K=L+I                                                             331Ø
            X=GRIDX(I+1)                                                      332Ø
            CORE(K)=PDERGH(X,W)                                               333Ø
  1Ø1    CONTINUE                                                           334Ø
  1Ø2 CONTINUE                                                              335Ø
      K=IRO                                                                   336Ø
C                                                                             337Ø
C     CORNER INDECIES                                                         338Ø
C                                                                             339Ø
      NR=IRO+NX                                                               34ØØ
      NU=IRO+NY*INC                                                           341Ø
      NRU=NU+NX                                                               342Ø
C                                                                             343Ø
      IF (ORDER.EQ.4) GO TO 1Ø8                                               344Ø
C                                                                             345Ø
C     INITIALIZATIONS                                                         346Ø
C                                                                             347Ø
      DO 1Ø3 I=IØ,NX                                                          348Ø
         CORE(I+1)=CORE(K)                                                    349Ø
         K=K+1                                                                35ØØ
  1Ø3 CONTINUE                                                              351Ø
C                                                                             352Ø
C     EVALUATE RIGHT SIDE AT THE EXTRA POINTS NEEDED BY SIX                   353Ø
C     ORDER FORMULA                                                           354Ø
C                                                                             355Ø
      YMIDL=-HY*.5                                                            356Ø
      DO 1Ø5 J=1,NY                                                           357Ø
         L=NX*(J-1)                                                           358Ø
         YMIDL=YMIDL+HY                                                       359Ø
         XMIDL=-HX*.5                                                         36ØØ
         DO 1Ø4 I=1,NX                                                        361Ø
            XMIDL=XMIDL+HX                                                    362Ø
            K=I+L                                                             363Ø
            PTINT(K)=PDERGH(XMIDL,YMIDL)                                      364Ø
  1Ø4    CONTINUE                                                           365Ø
  1Ø5 CONTINUE                                                              366Ø
C                                                                             367Ø
C     COMPUTE THE RIGHT SIDE OF SIX ORDER DIFFERENCE OPERATOR                 368Ø
C                                                                             369Ø
      LL=IRO                                                                  37ØØ
      DO 1Ø7 J=JMIN,JMAX                                                      371Ø
         LL=LL+INC                                                            372Ø
         L=LL                                                                 373Ø
         CNTRLF=CORE(L)                                                       374Ø
         LDOWN=L-INC                                                          375Ø
         DOWNLF=CORE(LDOWN)                                                   376Ø
         DO 1Ø6 I=IMIN,IMAX                                                   377Ø
            LUP=L+INC                                                         378Ø
            LUP2=LUP+2                                                        379Ø
            LUP1=LUP+1                                                        38ØØ
            IP1=I+1                                                           381Ø
            IP2=I+2                                                           382Ø
            LP2=L+2                                                           383Ø
            K=L+1                                                             384Ø
            TEMP=DOWNLF+CORE(LUP)+CORE(LUP2)+CORE(IP2)+4.*(CNTRLF+CORE(L      385Ø
     1      UP1)+CORE(LP2)+CORE(IP1))+148.*CORE(K)                            386Ø
C                                                                             387Ø
            CNTRLF=CORE(K)                                                    388Ø
            DOWNLF=CORE(IP1)                                                  389Ø
            CORE(I+1)=CORE(K)                                                 39ØØ
            IDWN1=I+NX*(J-1)                                                  391Ø
            IDWN2=IDWN1+1                                                     392Ø
C                                                                             393Ø
            IUP1=IDWN1+NX                                                     394Ø
            IUP2=IUP1+1                                                       395Ø
C                                                                             396Ø
            CORE(K)=FACTOR*(TEMP+48.*(PTINT(IDWN1)+PTINT(IDWN2)+PTINT(IU      397Ø
     1      P1)+PTINT(IUP2)))                                                 398Ø
            L=L+1                                                             399Ø
  1Ø6    CONTINUE                                                           4ØØØ
```

```
         LP1=L+1                                                          4Ø1Ø
         CORE(NX+1)=CORE(LP1)                                             4Ø2Ø
  1Ø7 CONTINUE                                                            4Ø3Ø
C                                                                         4Ø4Ø
      GO TO 112                                                           4Ø5Ø
  1Ø8 CONTINUE                                                            4Ø6Ø
C                                                                         4Ø7Ø
C     INITIALIZATION                                                      4Ø8Ø
C                                                                         4Ø9Ø
      DO 1Ø9 I=IMIN,IMAX                                                  41ØØ
         K=K+1                                                            411Ø
         CORE(I)=CORE(K)                                                  412Ø
  1Ø9 CONTINUE                                                            413Ø
C                                                                         414Ø
C     GENERATE RIGHT SIDE OF DIFFERENCE EQUATIONS                         415Ø
C                                                                         416Ø
      L=IRO                                                               417Ø
      DO 111 J=JMIN,JMAX                                                  418Ø
         L=L+INC                                                          419Ø
         K=L                                                              42ØØ
         XCENTR=CORE(K)                                                   421Ø
         KRIGHT=K+1                                                       422Ø
         DO 11Ø I=IMIN,IMAX                                               423Ø
            K=KRIGHT                                                      424Ø
            KRIGHT=K+1                                                    425Ø
            KUP=K+INC                                                     426Ø
            XLEFT=XCENTR                                                  427Ø
            XRIGHT=CORE(KRIGHT)                                           428Ø
            XVERT=CORE(I)+CORE(KUP)                                       429Ø
            XCENTR=CORE(K)                                                43ØØ
            CORE(I)=XCENTR                                                431Ø
            CORE(K)=FACTOR*(XVERT+HH*(XLEFT+XRIGHT)+CH*XCENTR)            432Ø
  11Ø    CONTINUE                                                         433Ø
  111 CONTINUE                                                            434Ø
C                                                                         435Ø
  112 CONTINUE                                                            436Ø
C                                                                         437Ø
C                                                                         438Ø
C     BOUNDARY VALUES AT THE CORNERS                                      439Ø
C                                                                         44ØØ
      CORE(IRO)=BCOND(3,Ø.,Ø.,BVALUS)                                     441Ø
      CORE(NR)=BCOND(2,SX,Ø.,BVALUS)                                      442Ø
      CORE(NU)=BCOND(4,Ø.,SY,BVALUS)                                      443Ø
      CORE(NRU)=BCOND(1,SX,SY,BVALUS)                                     444Ø
C                                                                         445Ø
C     ENFORCE DIRICHLET BOUNDARY CONDITIONS                               446Ø
C                                                                         447Ø
      KLEFT=IRO                                                           448Ø
      KLFTUP=IRO+INC                                                      449Ø
      MRIGHT=NR                                                           45ØØ
      MRGTUP=MRIGHT+INC                                                   451Ø
      CORE(KLFTUP)=BCOND(3,Ø.,HY,BVALUS)                                  452Ø
      CORE(MRGTUP)=BCOND(1,SX,HY,BVALUS)                                  453Ø
      DO 113 J=JMIN,JMAX                                                  454Ø
         W=GRIDY(J+2)                                                     455Ø
         KLEFTD=KLEFT                                                     456Ø
         KLEFT=KLFTUP                                                     457Ø
         KLFTUP=KLFTUP+INC                                                458Ø
         K=KLEFT+1                                                        459Ø
         MRGTD=MRIGHT                                                     46ØØ
         MRIGHT=MRGTUP                                                    461Ø
         MRGTUP=MRGTUP+INC                                                462Ø
         M=MRIGHT-1                                                       463Ø
         CORE(KLFTUP)=BCOND(3,Ø.,W,BVALUS)                                464Ø
         CORE(MRGTUP)=BCOND(1,SX,W,BVALUS)                                465Ø
         CORE(K)=CORE(K)-POTFAC*(CORE(KLEFTD)+CORE(KLFTUP)+OFFD*CORE(KLE  466Ø
     1   FT))                                                             467Ø
         CORE(M)=CORE(M)-POTFAC*(CORE(MRGTD)+CORE(MRGTUP)+OFFD*CORE(MRIG  468Ø
     1   HT))                                                             469Ø
  113 CONTINUE                                                            47ØØ
      KDOWN=IRO                                                           471Ø
      KRGTD=IRO+1                                                         472Ø
      MUP=NU                                                              473Ø
      MRGTUP=NU+1                                                         474Ø
      CORE(KRGTD)=BCOND(2,HX,Ø.,BVALUS)                                   475Ø
      CORE(MRGTUP)=BCOND(4,HX,SY,BVALUS)                                  476Ø
```

```
      DO 114 I=IMIN,IMAX                                                4770
         W=GRIDX(I+2)                                                   4780
         KLEFTD=KDOWN                                                   4790
         KDOWN=KRGTD                                                    4800
         KRGTD=KRGTD+1                                                  4810
         MLFTUP=MUP                                                     4820
         MUP=MRGTUP                                                     4830
         MRGTUP=MRGTUP+1                                                4840
         K=KDOWN+INC                                                    4850
         M=MUP-INC                                                      4860
         CORE(KRGTD)=BCOND(2,W,0.,BVALUS)                               4870
         CORE(MRGTUP)=BCOND(4,W,SY,BVALUS)                              4880
         CORE(K)=CORE(K)-POTFAC*(CORE(KLEFTD)+CORE(KRGTD)+DLEFT*CORE(KDO 4890
     1   WN))                                                           4900
         CORE(M)=CORE(M)-POTFAC*(CORE(MLFTUP)+CORE(MRGTUP)+DLEFT*CORE(MU 4910
     1   P))                                                            4920
  114 CONTINUE                                                          4930
C                                                                       4940
C     CORRECT CORNERS OF RECTANGLE                                      4950
C                                                                       4960
      K=IRO+INC+1                                                       4970
      KR=K+NX-2                                                         4980
      KU=NU-INC+1                                                       4990
      KRU=KU+NX-2                                                       5000
      CORE(K)=CORE(K)+POTFAC*CORE(IRO)                                  5010
      CORE(KR)=CORE(KR)+POTFAC*CORE(NR)                                 5020
      CORE(KU)=CORE(KU)+POTFAC*CORE(NU)                                 5030
      CORE(KRU)=CORE(KRU)+POTFAC*CORE(NRU)                              5040
      RETURN                                                            5050
C                                                                       5060
      END                                                               5070
      SUBROUTINE EQSOL (CORE,NCORED,Z,NZD,SI,NSID,Y,NYD,INDEX,INDEXD,ED, 5080
     1NEDD,AKX,NAKXD)                                                   5090
C                                                                       5100
C     SOLVES THE BLOCK TRIDIAGONAL SYSTEM OF DIFFERENCE EQUATIONS       5110
C     USING FAST FOURIER SERIES METHOD                                  5120
C                                                                       5130
C     THE ARGUMENTS - CORE(NCORED),Z(NZD),SI(NSID),Y(NYD),INDEX(INDEXD), 5140
C                     ED(NEDD),AKX(NAKXD) - DEFINED IN FFT9 PROGRAM     5150
C                                                                       5160
      REAL CORE(NCORED),Z(NZD),SI(NSID),Y(NYD),ED(NEDD),AKX(NAKXD)      5170
      INTEGER INDEX(INDEXD)                                             5180
C                                                                       5190
C     FFT9 COMMON VARIABLES                                             5200
C                                                                       5210
      COMMON /MESH/ NX,NY,IMIN,IMAX,JMIN,JMAX,INC,IRO,IBCX,IQX,IBCY,IQY, 5220
     1HX,HY,HXY2,PI,POTFAC,SX,SY                                        5230
      COMMON /FDFORM/ DLEFT,DIAG,OFFD,RF,HH,CH,RC1,FACTOR               5240
      J1=2                                                              5250
      J2=NY-2                                                           5260
C                                                                       5270
C     MODIFICATION OF EVEN COLUMN VECTORS                               5280
C                                                                       5290
      CALL EVENRD (J1,J2,CORE,NCORED)                                   5300
C                                                                       5310
C     PERFORM FOURIER ANALYSIS ON EVEN LINES                            5320
C                                                                       5330
      K=IRO+INC*J1                                                      5340
      JUMP=INC+INC                                                      5350
      DO 101 J=J1,J2,2                                                  5360
         CALL FETCHX (K,Z,NZD,CORE,NCORED)                              5370
         CALL FOUR (IBCX,IQX,SI,NSID,Z,NZD,Y,NYD,INDEX,INDEXD)          5380
         CALL STOREX (K,CORE,NCORED,Y,NYD)                              5390
  101 K=K+JUMP                                                          5400
C                                                                       5410
C     DIVISION BY EIGENVALUES OF OFFDIAGONAL MATRIX                     5420
C                                                                       5430
      DO 103 I=IMIN,IMAX                                                5440
         M=IRO+I                                                        5450
         DO 102 J=J1,J2,2                                               5460
            K=M+INC*J                                                   5470
            CORE(K)=CORE(K)/ED(I+1)                                     5480
  102    CONTINUE                                                       5490
  103 CONTINUE                                                          5500
C                                                                       5510
C     SOLUTION FOR EVEN LINES BY CYCLIC REDUCTION                       5520
```

```
C                                                                       553Ø
      DO 1Ø4 K=IMIN,IMAX                                                554Ø
         A=AKX(K+1)                                                     555Ø
         L=IRO+K                                                        556Ø
         M=JUMP                                                         557Ø
  1Ø4 CALL CRED (IBCY,L,M,A,IQY-1,CORE,NCORED)                          558Ø
C                                                                       559Ø
C     FOURIER SYNTHESIS ON EVEN LINES                                   56ØØ
C                                                                       561Ø
      K=IRO+INC*J1                                                      562Ø
      DO 1Ø5 J=J1,J2,2                                                  563Ø
         CALL FETCHX (K,Z,NZD,CORE,NCORED)                              564Ø
         CALL FOUR (IBCX,IQX,SI,NSID,Z,NZD,Y,NYD,INDEX,INDEXD)          565Ø
         CALL STOREX (K,CORE,NCORED,Y,NYD)                              566Ø
  1Ø5 K=K+JUMP                                                          567Ø
C                                                                       568Ø
C     MODIFICATION OF ODD LINE VECTORS                                  569Ø
C                                                                       57ØØ
      J2=NY-1                                                           571Ø
      F1=1.Ø/POTFAC                                                     572Ø
      CALL ODDRD (F1,J2,CORE,NCORED)                                    573Ø
C                                                                       574Ø
C     SOLUTION FOR ODD LINES BY CYCLIC REDUCTION                        575Ø
C                                                                       576Ø
      A=-DIAG/OFFD                                                      577Ø
      DO 1Ø6 J=1,J2,2                                                   578Ø
         L=IRO+INC*J                                                    579Ø
  1Ø6 CALL CRED (IBCX,L,1,A,IQX,CORE,NCORED)                            58ØØ
C                                                                       581Ø
      RETURN                                                            582Ø
C                                                                       583Ø
      END                                                               584Ø
      SUBROUTINE DISCRT (ORDER,AKX,NAKXD,ED,NEDD,GRIDX,NGRDXD,GRIDY,NGRD 585Ø
     1YD)                                                               586Ø
C                                                                       587Ø
C     SETS CONSTANTS,CALCULATE GRID SPECIFICATIONS,9-POINT              588Ø
C     DIFFERENCE FORMULA AND THE EIGENVALUES OF THE DIAGONAL            589Ø
C     AND OFF DIAGONAL BLOCKS OF THE FINITE DIFFERENCE EQUATIONS         59ØØ
C                                                                       591Ø
C     THE ARGUMENTS-ORDER,AKX(NAKXD),ED(NEDD),GRIDX(NGRDXD),            592Ø
C                   GRIDY(NGRDYD) ARE DEFINED IN FFT9                   593Ø
C                                                                       594Ø
      INTEGER ORDER                                                     595Ø
      REAL AKX(NAKXD),ED(NEDD),GRIDX(NGRDXD),GRIDY(NGRDYD)              596Ø
C                                                                       597Ø
C     FFT9 COMMON VARIABLES                                             598Ø
C                                                                       599Ø
      COMMON /MESH/ NX,NY,IMIN,IMAX,JMIN,JMAX,INC,IRO,IBCX,IQX,IBCY,IQY, 6ØØØ
     1HX,HY,HXY2,PI,POTFAC,SX,SY                                        6Ø1Ø
      COMMON /FDFORM/ DLEFT,DIAG,OFFD,RF,HH,CH,RC1,FACTOR               6Ø2Ø
      COMMON /CPDE/ C1,C2,C3                                            6Ø3Ø
C                                                                       6Ø4Ø
C     GENERATE CONSTANTS                                                6Ø5Ø
C                                                                       6Ø6Ø
      PI=3.14159265358979                                               6Ø7Ø
      NX=2**IQX                                                         6Ø8Ø
      NY=2**IQY                                                         6Ø9Ø
      REV=1./FLOAT(NX)                                                  61ØØ
      REVY=1./FLOAT(NY)                                                 611Ø
      POTFAC=2.*REV                                                     612Ø
      HX=SX*REV                                                         613Ø
      HY=SY*REVY                                                        614Ø
      RF=PI*REV                                                         615Ø
C                                                                       616Ø
C     GENERATE GRID SPECIFICATIONS                                      617Ø
C                                                                       618Ø
      IØ=Ø                                                              619Ø
      DO 1Ø1 I=IØ,NX                                                    62ØØ
         GRIDX(I+1)=FLOAT(I)*HX                                         621Ø
  1Ø1 CONTINUE                                                          622Ø
      DO 1Ø2 J=IØ,NY                                                    623Ø
         GRIDY(J+1)=FLOAT(J)*HY                                         624Ø
  1Ø2 CONTINUE                                                          625Ø
      IMIN=1                                                            626Ø
      IMAX=NX-1                                                         627Ø
      JMIN=1                                                            628Ø
```

```
      JMAX=NY-1                                                          629Ø
      INC=IMAX-IMIN+3                                                    63ØØ
      IRO=2+INC                                                          631Ø
      HXY2=(HX/HY)**2                                                    632Ø
C                                                                        633Ø
C     INITIALIZATIONS                                                    634Ø
C                                                                        635Ø
      OFFD=4.                                                            636Ø
      DLEFT=4.                                                           637Ø
      DIAG=-2Ø.                                                          638Ø
      FACTOR=POTFAC*HX**2/6Ø.                                            639Ø
C                                                                        64ØØ
      IF (ORDER.EQ.6) GO TO 1Ø3                                          641Ø
C                                                                        642Ø
C     GENERATE COEFFICIENTS OF 9-POINT FINITE DIFFERENCE STENSIL         643Ø
C     OF FOURTH ORDER                                                    644Ø
C                                                                        645Ø
      RC1=1./C1                                                          646Ø
      C21=RC1*C2                                                         647Ø
      SIGMA=RC1*C3                                                       648Ø
      RGRID=HY/HX                                                        649Ø
      HX2=HX**2                                                          65ØØ
      SIGH2=SIGMA*HX2                                                    651Ø
      SIGH12=SIGH2/12.Ø                                                  652Ø
      RGRSQ=RGRID**2                                                     653Ø
      QUOT=RGRSQ/C21                                                     654Ø
      RR=1.-SIGH12                                                       655Ø
      QQ=C21*(1.-SIGH12*QUOT)/RR                                         656Ø
      DIV=RGRSQ+QQ                                                       657Ø
      OFFD=12.*QQ*QUOT/DIV-2.Ø                                           658Ø
      DLEFT=12.*C21/DIV-2.Ø                                              659Ø
      DIAG=(OFFD+2.)*RR*SIGH2-2.*(OFFD+DLEFT)-4.                         66ØØ
      HH=QQ/C21                                                          661Ø
      CH=(12.*RR-2.)*HH-2.                                               662Ø
      FACTOR=RC1*POTFAC*RGRSQ*HX2/DIV                                    663Ø
  1Ø3 CONTINUE                                                           664Ø
C                                                                        665Ø
C     CALCULATE EIGENVALUES OF THE OFF-DIAGONAL BLOCKS AND               666Ø
C     DIAGONAL BLOCKS                                                    667Ø
C                                                                        668Ø
      DO 1Ø4 I=IMIN,IMAX                                                 669Ø
         II=I+1                                                          67ØØ
         DFLI=FLOAT(I)                                                   671Ø
         TWOCOS=2.Ø*COS(RF*DFLI)                                         672Ø
         ED(II)=(DLEFT+TWOCOS)**2                                        673Ø
         RATIO=((DIAG+TWOCOS*OFFD)**2)/ED(II)                            674Ø
         AKX(II)=-2.Ø+RATIO                                              675Ø
  1Ø4 CONTINUE                                                           676Ø
      RETURN                                                             677Ø
C                                                                        678Ø
      END                                                                679Ø
      SUBROUTINE EVENRD (J1,J2,CORE,NCORED)                              68ØØ
C                                                                        681Ø
C     MODIFIES THE RIGHT SIDE ON EVEN LINE VECTORS WHERE J1 IS           682Ø
C     THE FIRST AND J2 THE LAST EVEN VECTOR.                             683Ø
C     THE RIGHT SIDE AND ITS MODIFICATION ARE STORED IN ARRAY CORE       684Ø
C                                                                        685Ø
C     THE ACTUAL ARGUMENT - CORE(NCORED) IS DEFINED IN FFT9              686Ø
C                                                                        687Ø
      REAL CORE(NCORED)                                                  688Ø
C                                                                        689Ø
C     FFT9 COMMON VARIABLES                                              69ØØ
C                                                                        691Ø
      COMMON /MESH/ NX,NY,IMIN,IMAX,JMIN,JMAX,INC,IRO,IBCX,IQX,IBCY,IQY, 692Ø
     1HX,HY,HXY2,PI,POTFAC,SX,SY                                         693Ø
      COMMON /FDFORM/ DLEFT,DIAG,OFFD,RF,HH,CH,RC1,FACTOR                694Ø
C                                                                        695Ø
C     INITIALIZATIONS                                                    696Ø
C                                                                        697Ø
      I1=IMIN                                                            698Ø
      I2=IMAX-1                                                          699Ø
      L=IRO                                                              7ØØ0
      JUMP=INC+INC                                                       7Ø1Ø
C                                                                        7Ø2Ø
      DO 1Ø2 J=J1,J2,2                                                   7Ø3Ø
         L=L+JUMP                                                        7Ø4Ø
```

```
            K=L                                                          7Ø5Ø
            KRIGHT=K+1                                                   7Ø6Ø
            KRDOWN=KRIGHT-INC                                            7Ø7Ø
            KRGTUP=KRIGHT+INC                                            7Ø8Ø
            X2=Ø.                                                        7Ø9Ø
            X3=CORE(KRDOWN)+CORE(KRGTUP)                                 71ØØ
            Y2=Ø.Ø                                                       711Ø
            Y3=CORE(KRIGHT)                                              712Ø
C                                                                        713Ø
            DO 1Ø1 I=I1,I2                                               714Ø
               K=K+1                                                     715Ø
               KRIGHT=KRIGHT+1                                           716Ø
               KRDOWN=KRDOWN+1                                           717Ø
               KRGTUP=KRGTUP+1                                           718Ø
               X1=X2                                                     719Ø
               X2=X3                                                     72ØØ
               X3=CORE(KRDOWN)+CORE(KRGTUP)                              721Ø
               Y1=Y2                                                     722Ø
               Y2=Y3                                                     723Ø
               Y3=CORE(KRIGHT)                                           724Ø
               CORE1=X1+X3+DLEFT*X2-OFFD*(Y1+Y3)-DIAG*Y2                 725Ø
               CORE(K)=CORE1                                             726Ø
  1Ø1      CONTINUE                                                      727Ø
            K=K+1                                                        728Ø
            CORE1=X2+DLEFT*X3-OFFD*Y2-DIAG*Y3                            729Ø
            CORE(K)=CORE1                                                73ØØ
  1Ø2   CONTINUE                                                         731Ø
        RETURN                                                           732Ø
C                                                                        733Ø
        END                                                              734Ø
        SUBROUTINE ODDRD (F1,J2,CORE,NCORED)                             735Ø
C                                                                        736Ø
C       MODIFIES THE RIGHT SIDE ON ODD-LINE VECTORS WHERE J2 IS THE LAST 737Ø
C       ODD LINE.THE RIGHT SIDE AND ITS MODIFICATION IS STORED IN        738Ø
C       ARRAY CORE(NCORED).                                              739Ø
C                                                                        74ØØ
C       THE ARGUMENT F1 IS A MULTIPLICATION FACTOR                       741Ø
C                                                                        742Ø
        REAL CORE(NCORED)                                                743Ø
C                                                                        744Ø
C       FFT9 COMMON VARIABLES                                            745Ø
C                                                                        746Ø
        COMMON /MESH/ NX,NY,IMIN,IMAX,JMIN,JMAX,INC,IRO,IBCX,IQX,IBCY,IQY, 747Ø
       1HX,HY,HXY2,PI,POTFAC,SX,SY                                       748Ø
        COMMON /FDFORM/ DLEFT,DIAG,OFFD,RF,HH,CH,RC1,FACTOR              749Ø
C                                                                        75ØØ
        DENOM=1.Ø/OFFD                                                   751Ø
        DLFT=DLEFT*DENOM                                                 752Ø
        CENTER=F1*DENOM                                                  753Ø
        I2=IMAX-1                                                        754Ø
        L=IRO-INC                                                        755Ø
        JUMP=INC+INC                                                     756Ø
C                                                                        757Ø
        DO 1Ø2 J=1,J2,2                                                  758Ø
           L=L+JUMP                                                      759Ø
           K=L                                                           76ØØ
           KRIGHT=K+1                                                    761Ø
           KRDOWN=KRIGHT-INC                                             762Ø
           KRGTUP=KRIGHT+INC                                             763Ø
           X2=Ø.Ø                                                        764Ø
           X3=CORE(KRDOWN)+CORE(KRGTUP)                                  765Ø
           IF (J.EQ.1) X3=CORE(KRGTUP)                                   766Ø
           IF (J.EQ.J2) X3=CORE(KRDOWN)                                  767Ø
           DO 1Ø1 I=IMIN,I2                                              768Ø
              K=K+1                                                      769Ø
              KRIGHT=KRIGHT+1                                            77ØØ
              KRDOWN=KRDOWN+1                                            771Ø
              KRGTUP=KRGTUP+1                                            772Ø
              X1=X2                                                      773Ø
              X2=X3                                                      774Ø
              X3=CORE(KRDOWN)+CORE(KRGTUP)                               775Ø
              IF (J.EQ.1) X3=CORE(KRGTUP)                                776Ø
              IF (J.EQ.J2) X3=CORE(KRDOWN)                               777Ø
              CORE1=CENTER*CORE(K)-DENOM*(X1+X3)-DLFT*X2                 778Ø
              CORE(K)=CORE1                                              779Ø
  1Ø1      CONTINUE                                                      78ØØ
```

```
C                                                                       781Ø
         K=K+1                                                          782Ø
         CORE1=CENTER*CORE(K)-DENOM*X2-DLFT*X3                          783Ø
         CORE(K)=CORE1                                                  784Ø
  1Ø2 CONTINUE                                                          785Ø
      RETURN                                                            786Ø
C                                                                       787Ø
      END                                                               788Ø
      SUBROUTINE CRED (IBC,L,M,A,IP1,CORE,NCORED)                       789Ø
C                                                                       79ØØ
C     SOLVES TRIANGULAR SYSTEMS BY RECURSIVE CYCLIC REDUCTION           791Ø
C     SEE REFERENCE @3                                                  792Ø
C                                                                       793Ø
C      IBC = 1,IP1-EXPONENT OF 2,A-DIAGONAL ELEMENT,L,M-MESH DEPENDENT  794Ø
C      CONSTANTS USED TO RECOVER RIGHT SIDE FROM ARRAY CORE             795Ø
C                                                                       796Ø
      REAL CORE(NCORED)                                                 797Ø
      COMMON /MESH/ NX,NY,IMIN,IMAX,JMIN,JMAX,INC,IRO,IBCX,IQX,IBCY,IQY,798Ø
     1HX,HY,HXY2,PI,POTFAC,SX,SY                                        799Ø
      DIMENSION BB(11)                                                  8ØØØ
C                                                                       8Ø1Ø
      IP=IP1                                                            8Ø2Ø
      N2=M                                                              8Ø3Ø
      BB(1)=A                                                           8Ø4Ø
      B=A                                                               8Ø5Ø
      N4=Ø                                                              8Ø6Ø
      N=2**IP                                                           8Ø7Ø
      K=L+N*M                                                           8Ø8Ø
      IP=IP-1                                                           8Ø9Ø
C                                                                       81ØØ
      DO 1Ø4 N1=1,IP                                                    811Ø
         N4=N4+1                                                        812Ø
         N3=N2                                                          813Ø
         N2=N2+N2                                                       814Ø
         J1=N2+L                                                        815Ø
         J3=K-N3                                                        816Ø
         J2=K-N2                                                        817Ø
         IF (J1.GT.J2) GO TO 1Ø2                                        818Ø
C                                                                       819Ø
         DO 1Ø1 J=J1,J2,N2                                              82ØØ
            JMN3=J-N3                                                   821Ø
            JPN3=J+N3                                                   822Ø
            CORE(J)=B*CORE(J)+CORE(JMN3)+CORE(JPN3)                     823Ø
  1Ø1    CONTINUE                                                       824Ø
  1Ø2    B=B*B-2.ØØØ                                                    825Ø
         BB(N1+1)=B                                                     826Ø
         IF (B.LE.1.ØE14) GO TO 1Ø4                                     827Ø
C                                                                       828Ø
C     SHORT CUT AND SOLVE BY DIVISION IF B                              829Ø
C     LARGER THAN DESIRED ACCURACY                                      83ØØ
C                                                                       831Ø
         IF (J1.GT.J2) GO TO 1Ø5                                        832Ø
C                                                                       833Ø
         DO 1Ø3 J=J1,J2,N2                                              834Ø
  1Ø3    CORE(J)=-CORE(J)/B                                             835Ø
C                                                                       836Ø
         IF (IBC.EQ.1) GO TO 1Ø5                                        837Ø
  1Ø4 CONTINUE                                                          838Ø
      I=L+N*M/2                                                         839Ø
      CORE(I)=-CORE(I)/B                                                84ØØ
C                                                                       841Ø
  1Ø5 DO 1Ø8 NN=1,N4                                                    842Ø
         N1=N4-NN                                                       843Ø
         B=BB(N1+1)                                                     844Ø
         J2=K-N3                                                        845Ø
         J1=L+N3                                                        846Ø
         J1PN3=J1+N3                                                    847Ø
         CORE(J1)=(CORE(J1PN3)-CORE(J1))/B                              848Ø
         J2MN3=J2-N3                                                    849Ø
         CORE(J2)=(CORE(J2MN3)-CORE(J2))/B                              85ØØ
         J1=J1+N2                                                       851Ø
         J2=J2-N2                                                       852Ø
         IF (J1.GT.J2) GO TO 1Ø7                                        853Ø
C                                                                       854Ø
         DO 1Ø6 J=J1,J2,N2                                              855Ø
            JMN3=J-N3                                                   856Ø
```

```
            JPN3=J+N3                                                       857Ø
            CORE(J)=(CORE(JMN3)+CORE(JPN3)-CORE(J))/B                       858Ø
  1Ø6     CONTINUE                                                          859Ø
  1Ø7     N2=N3                                                             86ØØ
  1Ø8 N3=N3/2                                                               861Ø
      RETURN                                                                862Ø
C                                                                           863Ø
      END                                                                   864Ø
      SUBROUTINE STOREX (K,CORE,NCORED,Y,NYD)                               865Ø
C                                                                           866Ø
C     TRANSFERS DATA FROM ARRAY Y TO ARRAY CORE                             867Ø
C     AFTER FOURIER ANALYSIS                                                868Ø
C                                                                           869Ø
C     THE ARGUMENTS - K IS THE NODE,CORE(NCORED),Y(NYD) DEFINED IN FFT9     87ØØ
C                                                                           871Ø
      REAL CORE(NCORED),Y(NYD)                                              872Ø
      COMMON /MESH/ NX,NY,IMIN,IMAX,JMIN,JMAX,INC,IRO,IBCX,IQX,IBCY,IQY,    873Ø
     1HX,HY,HXY2,PI,POTFAC,SX,SY                                            874Ø
      DO 1Ø1 I=IMIN,IMAX                                                    875Ø
         KPI=K+I                                                            876Ø
         IP1=I+1                                                            877Ø
         CORE(KPI)=Y(IP1)                                                   878Ø
  1Ø1 CONTINUE                                                              879Ø
      RETURN                                                                88ØØ
C                                                                           881Ø
      END                                                                   882Ø
      SUBROUTINE FETCHX (K,Z,NZD,CORE,NCORED)                               883Ø
C                                                                           884Ø
C     TRANSFERS DATA FROM ARRAY CORE TO ARRAY Z PRIOR TO                    885Ø
C     FOURIER ANALYSIS                                                      886Ø
C                                                                           887Ø
C     THE ARGUMENTS - K  NODE,Z(NZD),CORE(NCORED) ARE DEFINED IN FFT9       888Ø
C                                                                           889Ø
      REAL Z(NZD),CORE(NCORED)                                              89ØØ
      COMMON /MESH/ NX,NY,IMIN,IMAX,JMIN,JMAX,INC,IRO,IBCX,IQX,IBCY,IQY,    891Ø
     1HX,HY,HXY2,PI,POTFAC,SX,SY                                            892Ø
      DO 1Ø1 I=IMIN,IMAX                                                    893Ø
         KPI=K+I                                                            894Ø
         IP1=I+1                                                            895Ø
         Z(IP1)=CORE(KPI)                                                   896Ø
  1Ø1 CONTINUE                                                              897Ø
      RETURN                                                                898Ø
C                                                                           899Ø
      END                                                                   9ØØØ
      SUBROUTINE KFOLD (INDEX,INDEXD,SI,NSID,Y,NYD,Z,NZD)                   9Ø1Ø
C                                                                           9Ø2Ø
C     EVALUATES THE SUMMATIONS AND DOES ALL THE MULTIPLICATIONS             9Ø3Ø
C     BY A RECURSIVE TECHNIQUE (SEE REFERENCE @3 )                          9Ø4Ø
C                                                                           9Ø5Ø
C     THE ARGUMENTS-INDEX(INDEXD),SI(NSID),Y(NYD),Z(NZD) DEFINED IN FFT9    9Ø6Ø
C                                                                           9Ø7Ø
      REAL SI(NSID),Y(NYD),Z(NZD)                                           9Ø8Ø
      INTEGER INDEX(INDEXD)                                                 9Ø9Ø
C                                                                           91ØØ
C     FFT9 COMMON VARIABLES                                                 911Ø
C                                                                           912Ø
      COMMON /FFT/ N2,N4,N3,N7,IP,ISL,L1,IBCJ                               913Ø
C                                                                           914Ø
      JS1=N2                                                                915Ø
      I=1                                                                   916Ø
      J5=ISL+N2                                                             917Ø
      IS1=ISL                                                               918Ø
      IC1=L1                                                                919Ø
      JS1=JS1/2                                                             92ØØ
C                                                                           921Ø
C     GO TO FIRST TIME IS LAST TIME                                         922Ø
C                                                                           923Ø
      IF (JS1.EQ.1) GO TO 1Ø6                                               924Ø
      SN=SI(1)                                                              925Ø
      IS1=IS1+JS1                                                           926Ø
      IC1=IC1+JS1                                                           927Ø
      J3=IS1+JS1                                                            928Ø
C                                                                           929Ø
  1Ø1 IS0=IS1-JS1                                                           93ØØ
      IC0=IC1-JS1                                                           931Ø
      ODD1=SN*(Z(IC1)-Z(IS1))                                               932Ø
```

```
      ODD2=SN*(Z(IC1)+Z(IS1))                                    933Ø
      Z(IC1)=Z(ICO)-ODD1                                         934Ø
      Z(ICO)=Z(ICO)+ODD1                                         935Ø
      Z(IS1)=-Z(ISO)+ODD2                                        936Ø
      Z(ISO)=Z(ISO)+ODD2                                         937Ø
      IS1=IS1+1                                                  938Ø
      IC1=IC1+1                                                  939Ø
      IF (IS1.NE.J3) GO TO 1Ø1                                   94ØØ
C                                                                941Ø
      I=I+1                                                      942Ø
 1Ø2  IS1=ISL                                                    943Ø
      IC1=L1                                                     944Ø
      JS1=JS1/2                                                  945Ø
C                                                                946Ø
C     GO TO LAST TIME WITH K IN PAIRS                            947Ø
C                                                                948Ø
      IF (JS1.EQ.1) GO TO 1Ø7                                    949Ø
C                                                                95ØØ
C     TAKE K IN PAIRS INTERCHANGING SN AND CS                    951Ø
C                                                                952Ø
 1Ø3  SN=SI(I)                                                   953Ø
      I=I+1                                                      954Ø
      CS=SI(I)                                                   955Ø
      IS1=IS1+JS1                                                956Ø
      IC1=IC1+JS1                                                957Ø
      J3=IS1+JS1                                                 958Ø
C                                                                959Ø
 1Ø4  ISO=IS1-JS1                                                96ØØ
      ICO=IC1-JS1                                                961Ø
      ODD1=CS*Z(IC1)-SN*Z(IS1)                                   962Ø
      ODD2=SN*Z(IC1)+CS*Z(IS1)                                   963Ø
      Z(IC1)=Z(ICO)-ODD1                                         964Ø
      Z(ICO)=Z(ICO)+ODD1                                         965Ø
      Z(IS1)=-Z(ISO)+ODD2                                        966Ø
      Z(ISO)=Z(ISO)+ODD2                                         967Ø
      IS1=IS1+1                                                  968Ø
      IC1=IC1+1                                                  969Ø
      IF (IS1.NE.J3) GO TO 1Ø4                                   97ØØ
C                                                                971Ø
      IS1=IS1+JS1                                                972Ø
      IC1=IC1+JS1                                                973Ø
      J3=IS1+JS1                                                 974Ø
C                                                                975Ø
 1Ø5  ISO=IS1-JS1                                                976Ø
      ICO=IC1-JS1                                                977Ø
      ODD1=SN*Z(IC1)-CS*Z(IS1)                                   978Ø
      ODD2=CS*Z(IC1)+SN*Z(IS1)                                   979Ø
      Z(IC1)=Z(ICO)-ODD1                                         98ØØ
      Z(ICO)=Z(ICO)+ODD1                                         981Ø
      Z(IS1)=-Z(ISO)+ODD2                                        982Ø
      Z(ISO)=Z(ISO)+ODD2                                         983Ø
      IS1=IS1+1                                                  984Ø
      IC1=IC1+1                                                  985Ø
      IF (IS1.NE.J3) GO TO 1Ø5                                   986Ø
      I=I+1                                                      987Ø
      IF (IS1.EQ.J5) GO TO 1Ø2                                   988Ø
      GO TO 1Ø3                                                  989Ø
C                                                                99ØØ
C     LAST TIME IS FIRST TIME                                    991Ø
C                                                                992Ø
 1Ø6  K1=INDEX(I)                                                993Ø
      SN=SI(I)                                                   994Ø
      ISO=IS1                                                    995Ø
      IS1=IS1+JS1                                                996Ø
      ICO=IC1                                                    997Ø
      IC1=IC1+JS1                                                998Ø
      ODD1=SN*(Z(IC1)-Z(IS1))                                    999Ø
      Y(K1+1)=Z(ICO)+ODD1                                       1ØØØØ
      NDUM=N3-K1+1                                              1ØØ1Ø
      Y(NDUM)=Z(ICO)-ODD1                                       1ØØ2Ø
      RETURN                                                    1ØØ3Ø
C                                                               1ØØ4Ø
C     LAST TIME TAKING K IN PAIRS                               1ØØ5Ø
C                                                               1ØØ6Ø
 1Ø7  K1=INDEX(I)                                               1ØØ7Ø
      SN=SI(I)                                                  1ØØ8Ø
```

```
      I=I+1                                                          1ØØ9Ø
      CS=SI(I)                                                       1Ø1ØØ
      ISO=IS1                                                        1Ø11Ø
      IS1=IS1+JS1                                                    1Ø12Ø
      ICO=IC1                                                        1Ø13Ø
      IC1=IC1+JS1                                                    1Ø14Ø
      ODD1=CS*Z(IC1)-SN*Z(IS1)                                       1Ø15Ø
      Y(K1+1)=Z(ICO)+ODD1                                            1Ø16Ø
      NDUM=N3-K1+1                                                   1Ø17Ø
      Y(NDUM)=Z(ICO)-ODD1                                            1Ø18Ø
C                                                                    1Ø19Ø
      IS1=IS1+1                                                      1Ø2ØØ
      IC1=IC1+1                                                      1Ø21Ø
      K1=INDEX(I)                                                    1Ø22Ø
      ISO=IS1                                                        1Ø23Ø
      IS1=IS1+JS1                                                    1Ø24Ø
      ICO=IC1                                                        1Ø25Ø
      IC1=IC1+JS1                                                    1Ø26Ø
      ODD1=SN*Z(IC1)-CS*Z(IS1)                                       1Ø27Ø
      Y(K1+1)=Z(ICO)+ODD1                                            1Ø28Ø
      NDUM=N3-K1+1                                                   1Ø29Ø
      Y(NDUM)=Z(ICO)-ODD1                                            1Ø3ØØ
C                                                                    1Ø31Ø
      IS1=IS1+1                                                      1Ø32Ø
      IC1=IC1+1                                                      1Ø33Ø
      I=I+1                                                          1Ø34Ø
      IF (IS1.NE.J5) GO TO 1Ø7                                       1Ø35Ø
      RETURN                                                         1Ø36Ø
C                                                                    1Ø37Ø
      END                                                            1Ø38Ø
      SUBROUTINE FOUR (IBC1,IQ1,SI,NSID,Z,NZD,Y,NYD,INDEX,INDEXD)    1Ø39Ø
C                                                                    1Ø4ØØ
C     PERFORMS A SINE ANALYSIS OR SYNTHESIS ON THE INPUT             1Ø41Ø
C      ARRAY Z(I) , I = 2,N AND PUTS THE RESULTS IN THE ARRAY Y      1Ø42Ø
C                                                                    1Ø43Ø
C      THE ARGUMENTS -  ALL - DEFINED IN FFT9 MAIN PROGRAM           1Ø44Ø
C                                                                    1Ø45Ø
      REAL SI(NSID),Z(NZD),Y(NYD)                                    1Ø46Ø
      INTEGER INDEX(INDEXD)                                          1Ø47Ø
C                                                                    1Ø48Ø
C     FFT9 COMMON VARIABLES                                          1Ø49Ø
C                                                                    1Ø5ØØ
      COMMON /FFT/ N2,N4,N3,N7,IP,IS1,L1,IBCK                        1Ø51Ø
C                                                                    1Ø52Ø
      IBC=IBC1                                                       1Ø53Ø
      IQ=IQ1                                                         1Ø54Ø
      A5=SI(1)                                                       1Ø55Ø
      N4=2**IQ                                                       1Ø56Ø
      N3=N4                                                          1Ø57Ø
      N5=N3/4                                                        1Ø58Ø
      N7=N3/2                                                        1Ø59Ø
      N11=3*N7                                                       1Ø6ØØ
      N31=N3+1                                                       1Ø61Ø
      Z(N31)=Ø.ØØØ                                                   1Ø62Ø
      Z(1)=Ø.ØØØ                                                     1Ø63Ø
      N2=N3                                                          1Ø64Ø
C                                                                    1Ø65Ø
      DO 1Ø1 I=2,IQ                                                  1Ø66Ø
C                                                                    1Ø67Ø
         CALL TFOLD (1,1,Z,NZD)                                      1Ø68Ø
  1Ø1 N2=N2/2                                                        1Ø69Ø
      Y(N7+1)=Z(2)                                                   1Ø7ØØ
      JF=N5                                                          1Ø71Ø
C                                                                    1Ø72Ø
      DO 1Ø2 J=2,IQ                                                  1Ø73Ø
         L1=N2+1                                                     1Ø74Ø
C                                                                    1Ø75Ø
         CALL ZERO (1,Z,NZD)                                         1Ø76Ø
         IS1=1                                                       1Ø77Ø
C                                                                    1Ø78Ø
         CALL KFOLD (INDEX,INDEXD,SI,NSID,Y,NYD,Z,NZD)               1Ø79Ø
         I1=3*JF+1                                                   1Ø8ØØ
         I2=4*JF                                                     1Ø81Ø
         I3=I1+(N2/2-1)*I2                                           1Ø82Ø
C                                                                    1Ø83Ø
         CALL NEG (I1,I3,I2,Y,NYD)                                   1Ø84Ø
```

```
         N2=N2+N2                                                    1Ø85Ø
  1Ø2 JF=JF/2                                                        1Ø86Ø
      RETURN                                                         1Ø87Ø
C                                                                    1Ø88Ø
      END                                                            1Ø89Ø
      SUBROUTINE NEG (I1,I3,I2,Y,NYD)                                1Ø9ØØ
C                                                                    1Ø91Ø
C     SETS THE Y(K) K=I1,I3,I2 EQUAL TO -Y(K)                        1Ø92Ø
C                                                                    1Ø93Ø
C     THE ARGUMENT Y(NYD) IS DEFINED IN FFT9                         1Ø94Ø
C                                                                    1Ø95Ø
      REAL Y(NYD)                                                    1Ø96Ø
      DO 1Ø1 K=I1,I3,I2                                              1Ø97Ø
  1Ø1 Y(K)=-Y(K)                                                     1Ø98Ø
      RETURN                                                         1Ø99Ø
C                                                                    11ØØØ
      END                                                            11Ø1Ø
      SUBROUTINE ZERO (L,Z,NZD)                                      11Ø2Ø
C                                                                    11Ø3Ø
C     SETS THE ELEMENTS OF Z(LM)=Ø , LM=L+I-1,FOR I=1,N2             11Ø4Ø
C                                                                    11Ø5Ø
      REAL Z(NZD)                                                    11Ø6Ø
      COMMON /FFT/ N2,N4,N3,N7,IP,ISL,L1,IBCM                        11Ø7Ø
      DO 1Ø1 I=1,N2                                                  11Ø8Ø
         LDUM=L+I-1                                                  11Ø9Ø
         Z(LDUM)=Ø.Ø                                                 111ØØ
  1Ø1 CONTINUE                                                       1111Ø
      RETURN                                                         1112Ø
C                                                                    1113Ø
      END                                                            1114Ø
      SUBROUTINE TFOLD (IS,L,Z,NZD)                                  1115Ø
C                                                                    1116Ø
C     FOLDS THE INPUT ARRAY Z(NZD) DEFINED IN FFT9                   1117Ø
C                                                                    1118Ø
      REAL Z(NZD)                                                    1119Ø
      COMMON /FFT/ N2,N4,N3,N7,IP,ISL,L1,IBCN                        112ØØ
      IH2=N2/2-1                                                     1121Ø
      DO 1Ø1 I=IS,IH2                                                1122Ø
         I1=I+L                                                      1123Ø
         I2=N2-I+L                                                   1124Ø
         A=Z(I1)                                                     1125Ø
         Z(I1)=A-Z(I2)                                               1126Ø
  1Ø1 Z(I2)=A+Z(I2)                                                  1127Ø
      RETURN                                                         1128Ø
C                                                                    1129Ø
      END                                                            113ØØ
      SUBROUTINE SETF (IBC1,IQ1,INDEX,INDEXD,SI,NSID)                1131Ø
C                                                                    1132Ø
C     INITIALIZES ARRAYS SI ,INDEX FOR THE USE                       1133Ø
C     BY FOUR( SEE REFERENCE @3 )                                    1134Ø
C                                                                    1135Ø
C     THE ARGUMENTS - ALL - DEFINED IN FFT9 MAIN PROGRAM             1136Ø
C                                                                    1137Ø
      REAL SI(NSID)                                                  1138Ø
      INTEGER INDEX(INDEXD)                                          1139Ø
      COMMON /FFT/ N2,N4,N3,N7,IP,ISL,L1,IBCO                        114ØØ
      DPI=3.14159265358979                                           1141Ø
      IBC=IBC1                                                       1142Ø
      IQ=IQ1                                                         1143Ø
      N3=2**IQ                                                       1144Ø
      N7=N3/2                                                        1145Ø
      N5=N3/4                                                        1146Ø
      I=1                                                            1147Ø
      INDEX(I)=N5                                                    1148Ø
      SI(I)=1.Ø/SQRT(2.Ø)                                            1149Ø
      K=1                                                            115ØØ
      I=I+1                                                          1151Ø
  1Ø1 IL=I                                                           1152Ø
      IF (I.EQ.N7) GO TO 1Ø3                                         1153Ø
  1Ø2 K1=INDEX(K)/2                                                  1154Ø
      INDEX(I)=K1                                                    1155Ø
      RK1=K1                                                         1156Ø
      RN3=N3                                                         1157Ø
      SI(I)=SIN(DPI*RK1/RN3)                                         1158Ø
      K1=N7-K1                                                       1159Ø
      I=I+1                                                          116ØØ
```

```
      INDEX(I)=K1                                                       11161Ø
      RK1=K1                                                            11162Ø
      RN3=N3                                                            11163Ø
      SI(I)=SIN(DPI*RK1/RN3)                                            11164Ø
      K=K+1                                                             11165Ø
      I=I+1                                                             11166Ø
      IF (K.NE.IL) GO TO 1Ø2                                            11167Ø
      GO TO 1Ø1                                                         11168Ø
 1Ø3 RETURN                                                             11169Ø
C                                                                       11170Ø
      END                                                               11171Ø
C ************** OUTPUT MODULU ********************                   11172Ø
C                                                                       11173Ø
      SUBROUTINE SUMARY (LEVEL,GRIDX,NGRDXD,GRIDY,NGRDYD,CORE,NCORED)   11174Ø
C                                                                       11175Ø
C     PRINTS THE COMPUTED SOLUTION AND MAX. ERROR,MAX.                  11176Ø
C     RELATIVE ERROR IF THE SOLUTION IS KNOWN                           11177Ø
C     THE ARGUMENTS - ALL - DEFINED IN FFT9 MAIN PROGRAM                11178Ø
C                                                                       11179Ø
C                                                                       11180Ø
C     COMMON VARIABLES OF FFT9                                          11181Ø
C                                                                       11182Ø
      REAL GRIDX(NGRDXD),GRIDY(NGRDYD),CORE(NCORED)                     11183Ø
      COMMON /MESH/ NX,NY,IMIN,IMAX,JMIN,JMAX,INC,IRO,IBCX,IQX,IBCY,IQY,11184Ø
     1HX,HY,HXY2,PI,POTFAC,SX,SY                                        11185Ø
      COMMON /CPDE/ CUXX,CUYY,CU                                        11186Ø
C                                                                       11187Ø
C                                                                       11188Ø
      MSLINX=NX+1                                                       11189Ø
      MSLINY=NY+1                                                       11190Ø
C                                                                       11191Ø
      WRITE (6,1Ø8) MSLINX,MSLINY                                       11192Ø
      IF (LEVEL.GT.Ø) GO TO 1Ø3                                         11193Ø
      WRITE (6,1Ø9)                                                     11194Ø
      DO 1Ø2 I=IMIN,IMAX                                                11195Ø
         DO 1Ø1 J=JMIN,JMAX                                             11196Ø
            K=IRO+I+INC*J                                               11197Ø
            WRITE (6,11Ø) GRIDX(I+1),GRIDY(J+1),CORE(K)                 11198Ø
 1Ø1     CONTINUE                                                       11199Ø
 1Ø2 CONTINUE                                                           12ØØØ
C                                                                       12Ø1Ø
      RETURN                                                            12Ø2Ø
C                                                                       12Ø3Ø
C     COMPUTE MAX.ABSOLUTE AND MAX.RELATIVE ERROR                       12Ø4Ø
C                                                                       12Ø5Ø
 1Ø3 CONTINUE                                                           12Ø6Ø
      IF (LEVEL.EQ.2) WRITE (6,111)                                     12Ø7Ø
      ERRMAX=Ø.                                                         12Ø8Ø
      RELMAX=Ø.                                                         12Ø9Ø
      DO 1Ø7 I=IMIN,IMAX                                                121ØØ
         DO 1Ø6 J=JMIN,JMAX                                             1211Ø
            K=IRO+I+INC*J                                               1212Ø
            EXCT=TRUE(GRIDX(I+1),GRIDY(J+1))                            1213Ø
            ERROR=ABS(CORE(K)-EXCT)                                     1214Ø
            IF (ERROR.LE.ERRMAX) GO TO 1Ø4                              1215Ø
            ERRMAX=ERROR                                                1216Ø
            XMAX=GRIDX(I+1)                                             1217Ø
            YMAX=GRIDY(J+1)                                             1218Ø
 1Ø4        CONTINUE                                                    1219Ø
            IF (ABS(EXCT).LT.1.E-14) EXCT=1.E-14                        122ØØ
            RELERR=ERROR/ABS(EXCT)                                      1221Ø
            IF (RELERR.LE.RELMAX) GO TO 1Ø5                             1222Ø
            RELMAX=RELERR                                               1223Ø
            XREL=GRIDX(I+1)                                             1224Ø
            YREL=GRIDY(J+1)                                             1225Ø
 1Ø5        CONTINUE                                                    1226Ø
C                                                                       1227Ø
C     PRINT SOLUTION ,APPROXIMATE SOLUTION ,MAX.ERROR,REL.ERROR         1228Ø
C     AT EACH POINT                                                     1229Ø
C                                                                       123ØØ
            IF (LEVEL.EQ.2) WRITE (6,112) GRIDX(I+1),GRIDY(J+1),EXCT,COR 1231Ø
     1      E(K),ERROR,RELERR                                           1232Ø
 1Ø6    CONTINUE                                                       1233Ø
 1Ø7 CONTINUE                                                           1234Ø
      WRITE (6,113) ERRMAX,XMAX,YMAX,RELMAX,XREL,YREL                   1235Ø
      RETURN                                                            1236Ø
```

```
C                                                                       12370
  108 FORMAT (1X,22HGRID.      UNIFORM X =,I5,5X,11HUNIFORM Y =,I5)       12380
  109 FORMAT (1X,36HGRID POINT      APPROXIMATE SOLUTION)                 12390
  110 FORMAT (1X,2F8.3,12X,F10.5)                                         12400
  111 FORMAT (1X,45H GRID  POINT TRUE SOL.  APPR. SOL.  MAX.ERROR,6X,10H  12410
     1REL. ERROR)                                                       12420
  112 FORMAT (1X,2F5.2,2F10.5,2E15.3)                                   12430
  113 FORMAT (1X,14HMAX.ABS.ERROR=,E15.3,3X,3HAT ,F5.2,1H,,F5.2/1X,5HMAX  12440
     1.R,9HEL.ERROR=,E15.3,3X,2HAT,F5.2,1H,,F5.2)                       12450
C                                                                       12460
      END                                                               12470
C ***** PROBLEM DEFINITION SUBPROGRAMS FOR A TEST EXAMPLE *****         12480
C                                                                       12490
      REAL FUNCTIONPDERGH(X,Y)                                          12500
      PDERGH=6.*X*Y*EXP(X)*EXP(Y)*(X*Y+X+Y-3.)                          12510
      RETURN                                                            12520
C                                                                       12530
      END                                                               12540
      REAL FUNCTION BCOND(I,X,Y,BVALUS)                                 12550
      REAL BVALUS(4)                                                    12560
      BVALUS(4)=0.                                                      12570
      BCOND=BVALUS(4)                                                   12580
      RETURN                                                            12590
C                                                                       12600
      END                                                               12610
      FUNCTION TRUE(X,Y)                                                12620
      TRUE=3.*EXP(X)*EXP(Y)*(X-X*X)*(Y-Y*Y)                             12630
      RETURN                                                            12640
C                                                                       12650
      END                                                               12660
      SUBROUTINE PDE (X,Y,CVALUS)                                       12670
      REAL CVALUS(7)                                                    12680
C                                                                       12690
      CVALUS(1)=1.                                                      12700
      CVALUS(3)=1.                                                      12710
      CVALUS(6)=0.                                                      12720
      RETURN                                                            12730
C                                                                       12740
      END                                                               12750
                                                                        12760
1.        1.          9  9                                            12770
 1 5 4                                                                  12780
```

ALGORITHM 544
L2A and L2B, Weighted Least Squares Solutions by Modified Gram-Schmidt with Iterative Refinement [F4]

ROY H. WAMPLER
National Bureau of Standards

Key Words and Phrases: covariance matrix, curve fitting, iterative refinement, least squares solution, linear constraints, overdetermined system of equations, regression, underdetermined system of equations
CR Categories: 5.14, 5.5
Language: PFORT, a portable subset of ANSI Fortran

DESCRIPTION

The two algorithms given here are a complement to [1] where the types of problems which can be solved by L2A and L2B, together with the method of solution and details of the calling sequences, are described.

REFERENCES

1. WAMPLER, R.H. Solutions to weighted least squares problems by modified Gram-Schmidt with iterative refinement. *ACM Trans. Math. Software 5*, 4(Dec. 1979), 457-465.

ALGORITHM

```
C MAIN PROGRAM TO CALL SUBROUTINES L2A AND L2B FOR SOLVING LINEAR      DRV  1Ø
C LEAST SQUARES PROBLEMS.                                              DRV  2Ø
C                                                                      DRV  3Ø
C VERSION OF NOVEMBER 2, 1978.                                         DRV  4Ø
C                                                                      DRV  5Ø
C WRITTEN BY ROY H. WAMPLER, STATISTICAL ENGINEERING                   DRV  6Ø
C LABORATORY, NATIONAL BUREAU OF STANDARDS,                            DRV  7Ø
C WASHINGTON, D. C. 2Ø234.                                             DRV  8Ø
C                                                                      DRV  9Ø
C THE MAIN PROGRAM READS AND PRINTS THE INPUT DATA, CALLS EITHER       DRV  1ØØ
C SUBROUTINE L2A OR L2B, COMPUTES CERTAIN QUANTITIES DERIVED FROM      DRV  11Ø
C OUTPUT OF THE SUBROUTINE, AND PRINTS COMPUTED RESULTS.               DRV  12Ø
C                                                                      DRV  13Ø
C SEE COMMENTS AT THE BEGINNING OF SUBROUTINES L2A AND L2B FOR A       DRV  14Ø
C DESCRIPTION OF THE INPUT, OUTPUT AND INTERNAL VARIABLES WHICH APPEAR DRV  15Ø
C IN THE CALLS TO THOSE SUBROUTINES.                                   DRV  16Ø
C                                                                      DRV  17Ø
C MODE IS A PARAMETER WHICH CONTROLS WHETHER SUBROUTINE L2A OR         DRV  18Ø
C SUBROUTINE L2B SHALL BE CALLED BY THIS MAIN PROGRAM.  THE TWO        DRV  19Ø
C SUBROUTINES WILL FURNISH THE SAME SOLUTIONS WHENEVER THE COMPUTED    DRV  2ØØ
C RANK OF THE SYSTEM OF M EQUATIONS IN N UNKNOWNS EQUALS N AND M.GE.N. DRV  21Ø
C IN CASES WHERE THE COMPUTED RANK N1 IS LESS THAN N, THE USER         DRV  22Ø
C SPECIFIES THE TYPE OF SOLUTION TO BE COMPUTED ACCORDING TO WHETHER   DRV  23Ø
C MODE = 1 OR MODE = 2.                                                DRV  24Ø
C MATRIX A IS THE GIVEN MATRIX OF A SYSTEM OF M LINEAR EQUATIONS IN N  DRV  25Ø
C UNKNOWNS.  MATRIX W IS A GIVEN DIAGONAL MATRIX OF WEIGHTS WITH ALL   DRV  26Ø
```

Received 25 February 1976; revised 14 November 1978.
Author's address: National Bureau of Standards, Admin. A337, U. S. Department of Commerce, Washington, DC 20234.

```
C DIAGONAL ELEMENTS NONNEGATIVE.  LET H = (SQRT(W))*A.                  DRV  27Ø
C                                                                       DRV  28Ø
C MODE = 1 INDICATES THAT IF N1.LT.N THE ORIGINAL MATRIX H (M BY N)     DRV  29Ø
C          IS TO BE REPLACED BY A SMALLER MATRIX (M BY N1) WHOSE        DRV  3ØØ
C          COLUMNS ARE LINEARLY INDEPENDENT, AND A SOLUTION IS TO BE    DRV  31Ø
C          SOUGHT FOR THE SMALLER SYSTEM OF EQUATIONS.  THUS N - N1     DRV  32Ø
C          COLUMNS OF THE ORIGINAL MATRIX H ARE DELETED, AND            DRV  33Ø
C          COEFFICIENTS CORRESPONDING TO THESE N - N1 DELETED COLUMNS   DRV  34Ø
C          WILL BE SET EQUAL TO ZERO.                                   DRV  35Ø
C MODE = 2 INDICATES THAT A SOLUTION IS SOUGHT FOR A LEAST SQUARES      DRV  36Ø
C          PROBLEM HAVING N ELEMENTS IN THE SOLUTION VECTOR.  IN ORDER  DRV  37Ø
C          TO OBTAIN A UNIQUE SOLUTION, THE CONDITION THAT THE          DRV  38Ø
C          SOLUTION VECTOR BE OF MINIMAL EUCLIDEAN NORM IS IMPOSED.     DRV  39Ø
C                                                                       DRV  4ØØ
C THE SEQUENCE OF INPUT CARDS FOR THIS MAIN PROGRAM IS --               DRV  41Ø
C Ø. CARD SPECIFYING MODE DESIRED FOR PROBLEMS WHICH FOLLOW, IN (I5)    DRV  42Ø
C    FORMAT.                                                            DRV  43Ø
C 1. PROBLEM HEADING CARD, IN (8ØA1) FORMAT.                            DRV  44Ø
C 2. PARAMETER CARD IN (6I5,5X,F1Ø.Ø) FORMAT, GIVING VALUES OF THE      DRV  45Ø
C    PARAMETERS M, N, M1, L, ITYPE, IWGHT, TOL.                         DRV  46Ø
C    M      TOTAL NUMBER OF EQUATIONS.                                  DRV  47Ø
C    N      NUMBER OF UNKNOWN COEFFICIENTS.                             DRV  48Ø
C    M1     NUMBER OF LINEAR CONSTRAINTS (Ø.LE.M1.LE.M AND M1.LE.N).    DRV  49Ø
C    L      NUMBER OF RIGHT-HAND SIDES (VECTORS OF OBSERVATIONS).       DRV  5ØØ
C    ITYPE PARAMETER WHICH SPECIFIES WHETHER OR NOT DATA FOR A          DRV  51Ø
C          POLYNOMIAL TYPE FIT ARE TO BE READ IN.                       DRV  52Ø
C          ITYPE = 1 INDICATES POLYNOMIAL TYPE.                         DRV  53Ø
C          ITYPE = 2 INDICATES NON-POLYNOMIAL TYPE.                     DRV  54Ø
C    IWGHT PARAMETER WHICH SPECIFIES WHETHER OR NOT WEIGHTS ARE TO      DRV  55Ø
C          BE READ IN.                                                  DRV  56Ø
C          IWGHT = 1 INDICATES WEIGHTS ARE NOT TO BE READ IN.  (THE     DRV  57Ø
C                    PROGRAM SETS ALL WEIGHTS EQUAL TO 1.Ø.)            DRV  58Ø
C          IWGHT = 2 INDICATES WEIGHTS ARE TO BE READ IN.               DRV  59Ø
C    TOL   PARAMETER USED IN DETERMINING THE RANK OF MATRIX H.          DRV  6ØØ
C          NOTE --                                                      DRV  61Ø
C          (1) IF TOL EQUALS ZERO, THE TOLERANCE USED IN THE            DRV  62Ø
C              DECOMPOSITION SUBROUTINE WILL BE BASED ON MACHINE        DRV  63Ø
C              PRECISION.                                               DRV  64Ø
C          (2) IF TOL IS GREATER THAN ZERO, THIS VALUE OF TOL WILL BE   DRV  65Ø
C              USED IN SETTING AN ABSOLUTE TOLERANCE FOR COMPARISON     DRV  66Ø
C              WITH DIAGONAL ELEMENTS OF THE TRIANGULAR MATRIX OBTAINED DRV  67Ø
C              IN THE DECOMPOSITION SUBROUTINE.  THE VALUE OF TOL CAN   DRV  68Ø
C              BE BASED ON KNOWLEDGE CONCERNING THE ACCURACY OF THE     DRV  69Ø
C              DATA.                                                    DRV  7ØØ
C 3. CARD GIVING FORMAT OF THE DATA CARDS (CONTAINING A, B AND          DRV  71Ø
C    POSSIBLY W) WHICH FOLLOW.  THIS FORMAT CARD IS IN (8ØA1)           DRV  72Ø
C    FORMAT.                                                            DRV  73Ø
C 4. DATA CARDS FOR THE ARRAYS A, B AND POSSIBLY W.  THERE ARE FOUR     DRV  74Ø
C    POSSIBLE CONFIGURATIONS FOR THE DATA, DEPENDING ON THE VALUES      DRV  75Ø
C    OF ITYPE AND IWGHT.  (FOR POLYNOMIAL FITS, THE FIRST POWER OF A    DRV  76Ø
C    IS READ IN AND HIGHER POWERS ARE COMPUTED BY THE PROGRAM WHEN      DRV  77Ø
C    N.GT.2.)  THE FOUR CONFIGURATIONS ARE ILLUSTRATED BELOW BY         DRV  78Ø
C    SHOWING WHAT THE CARD (OR CARDS) FOR THE I-TH ROW OF DATA          DRV  79Ø
C    CONTAINS.                                                          DRV  8ØØ
C   A. ITYPE = 1, IWGHT = 1.                                            DRV  81Ø
C      POLYNOMIAL TYPE FIT.  EQUAL WEIGHTS, NOT TO BE READ IN.          DRV  82Ø
C     A(I,2)  B(I,1) B(I,2) ... B(I,L)                                  DRV  83Ø
C   B. ITYPE = 1, IWGHT = 2.                                            DRV  84Ø
C      POLYNOMIAL TYPE FIT.  UNEQUAL WEIGHTS, TO BE READ IN.            DRV  85Ø
C     A(I,2)  B(I,1) B(I,2) ... B(I,L)  W(I)                            DRV  86Ø
C   C. ITYPE = 2, IWGHT = 1.                                            DRV  87Ø
C      NON-POLYNOMIAL TYPE FIT.  EQUAL WEIGHTS, NOT TO BE READ IN.      DRV  88Ø
C     A(I,1) A(I,2) ... A(I,N)  B(I,1) B(I,2) ... B(I,L)                DRV  89Ø
C   D. ITYPE = 2, IWGHT = 2.                                            DRV  9ØØ
C      NON-POLYNOMIAL TYPE FIT.  UNEQUAL WEIGHTS, TO BE READ IN.        DRV  91Ø
C     A(I,1) A(I,2) ... A(I,N)  B(I,1) B(I,2) ... B(I,L)  W(I)          DRV  92Ø
C 5. CARD IN (I5) FORMAT GIVING VALUE OF THE PARAMETER IFDONE.  IF      DRV  93Ø
C    THE PROBLEM AT HAND IS TO BE FOLLOWED BY ANOTHER PROBLEM,          DRV  94Ø
C    IFDONE = 1.  OTHERWISE, IFDONE EQUALS ANY INTEGER EXCEPT 1.        DRV  95Ø
C                                                                       DRV  96Ø
C DIMENSIONS OF ARRAYS ARE SET ASSUMING THAT M.LE.21, N.LE.8, AND       DRV  97Ø
C L.LE.3.                                                               DRV  98Ø
C                                                                       DRV  99Ø
      INTEGER IPIVOT(8)                                                 DRV 1ØØØ
      REAL A(21,8), B(21,3), RES(21,3), TOL, W(21), X(8,3)              DRV 1Ø1Ø
C ARRAYS Q AND R ARE USED IN SUBROUTINE L2A.                            DRV 1Ø2Ø
```

```
C ARRAY QR IS USED IN SUBROUTINE L2B.                                    DRV 1Ø3Ø
      REAL Q(21,8), R(8,8)                                               DRV 1Ø4Ø
      REAL QR(29,8)                                                      DRV 1Ø5Ø
C SUBROUTINE L2A REQUIRES THAT C BE DIMENSIONED AT LEAST 4*(M+N) + 2*L.  DRV 1Ø6Ø
C SUBROUTINE L2B REQUIRES THAT C BE DIMENSIONED AT LEAST 6*(M+N) + 2*L.  DRV 1Ø7Ø
      REAL C(18Ø)                                                        DRV 1Ø8Ø
      REAL SD, SDX(8), SNORM, SS, Z                                      DRV 1Ø9Ø
      LOGICAL FAIL(3)                                                    DRV 11ØØ
      DIMENSION HEAD(8Ø), IFMT(8Ø)                                       DRV 111Ø
C                                                                        DRV 112Ø
C IN THE FOLLOWING DATA STATEMENT, NR IS THE CARD READER DEVICE          DRV 113Ø
C AND NW IS THE PRINTER DEVICE NUMBER.                                   DRV 114Ø
C                                                                        DRV 115Ø
      DATA NR,NW /5,6/                                                   DRV 116Ø
C                                                                        DRV 117Ø
C THE PARAMETERS MM, NN AND MMPNN WHICH APPEAR IN THE STATEMENTS         DRV 118Ø
C CALL L2A(...) AND CALL L2B(...) ARE USED IN SETTING ADJUSTABLE         DRV 119Ø
C DIMENSIONS OF ARRAYS.  THEY ARE GIVEN SPECIFIC VALUES IN THE           DRV 12ØØ
C FOLLOWING DATA STATEMENTS.  MMPNN IS USED ONLY IN CALL L2B(...).       DRV 121Ø
C                                                                        DRV 122Ø
      DATA MM,NN /21,8/                                                  DRV 123Ø
      DATA MMPNN /29/                                                    DRV 124Ø
C                                                                        DRV 125Ø
      READ (NR,9973Ø) MODE                                               DRV 126Ø
C                                                                        DRV 127Ø
C DEFAULT VALUE FOR MODE IS 1.                                           DRV 128Ø
C                                                                        DRV 129Ø
      IF (MODE.NE.2) MODE = 1                                            DRV 13ØØ
      IPROB = Ø                                                          DRV 131Ø
   1Ø READ (NR,9999Ø) HEAD                                               DRV 132Ø
      IPROB = IPROB + 1                                                  DRV 133Ø
      WRITE (NW,9998Ø) IPROB                                             DRV 134Ø
      WRITE (NW,9997Ø) HEAD                                              DRV 135Ø
      READ (NR,9996Ø) M,N,M1,L,ITYPE,IWGHT,TOL                           DRV 136Ø
      WRITE (NW,9995Ø)                                                   DRV 137Ø
      WRITE (NW,9994Ø) M,N,M1,L,ITYPE,IWGHT,MODE,TOL                     DRV 138Ø
      READ (NR,9999Ø) IFMT                                               DRV 139Ø
      WRITE (NW,9993Ø) IFMT                                              DRV 14ØØ
      GO TO (2Ø,1ØØ), ITYPE                                              DRV 141Ø
C                                                                        DRV 142Ø
C TYPE 1.  POLYNOMIAL FIT.                                               DRV 143Ø
   2Ø GO TO (3Ø,5Ø), IWGHT                                               DRV 144Ø
C      A.  EQUAL WEIGHTS, NOT TO BE READ IN.                             DRV 145Ø
   3Ø DO 4Ø I=1,M                                                        DRV 146Ø
        READ (NR,IFMT) A(I,2),(B(I,K),K=1,L)                             DRV 147Ø
        W(I) = 1.Ø                                                       DRV 148Ø
   4Ø CONTINUE                                                           DRV 149Ø
      GO TO 7Ø                                                           DRV 15ØØ
C      B.  UNEQUAL WEIGHTS, TO BE READ IN.                               DRV 151Ø
   5Ø DO 6Ø I=1,M                                                        DRV 152Ø
        READ (NR,IFMT) A(I,2),(B(I,K),K=1,L),W(I)                        DRV 153Ø
   6Ø CONTINUE                                                           DRV 154Ø
   7Ø DO 9Ø I=1,M                                                        DRV 155Ø
        A(I,1) = 1.Ø                                                     DRV 156Ø
        IF (N.LT.3) GO TO 9Ø                                             DRV 157Ø
        DO 8Ø J=3,N                                                      DRV 158Ø
          A(I,J) = A(I,2)**(J-1)                                         DRV 159Ø
   8Ø   CONTINUE                                                         DRV 16ØØ
   9Ø CONTINUE                                                           DRV 161Ø
      GO TO 15Ø                                                          DRV 162Ø
C                                                                        DRV 163Ø
C TYPE 2.  NON-POLYNOMIAL FIT.                                           DRV 164Ø
  1ØØ GO TO (11Ø,13Ø), IWGHT                                             DRV 165Ø
C      C.  EQUAL WEIGHTS, NOT TO BE READ IN.                             DRV 166Ø
  11Ø DO 12Ø I=1,M                                                       DRV 167Ø
        READ (NR,IFMT) (A(I,J),J=1,N),(B(I,K),K=1,L)                     DRV 168Ø
        W(I) = 1.Ø                                                       DRV 169Ø
  12Ø CONTINUE                                                           DRV 17ØØ
      GO TO 15Ø                                                          DRV 171Ø
C      D.  UNEQUAL WEIGHTS, TO BE READ IN.                               DRV 172Ø
  13Ø DO 14Ø I=1,M                                                       DRV 173Ø
        READ (NR,IFMT) (A(I,J),J=1,N),(B(I,K),K=1,L),W(I)                DRV 174Ø
  14Ø CONTINUE                                                           DRV 175Ø
  15Ø IF (M1.EQ.Ø .OR. IWGHT.EQ.1) GO TO 17Ø                             DRV 176Ø
C                                                                        DRV 177Ø
C INSURE THAT WEIGHTS EQUAL 1.Ø FOR THE FIRST M1 EQUATIONS WHEN M1       DRV 178Ø
```

```
C IS GREATER THAN ZERO.                                                 DRV 179Ø
C                                                                       DRV 18ØØ
      DO 16Ø I=1,M1                                                     DRV 181Ø
        W(I) = 1.Ø                                                      DRV 182Ø
  16Ø CONTINUE                                                          DRV 183Ø
C                                                                       DRV 184Ø
C PRINT A, B AND W.                                                     DRV 185Ø
C                                                                       DRV 186Ø
  17Ø WRITE (NW,9992Ø)                                                  DRV 187Ø
      KZ = Ø                                                            DRV 188Ø
      DO 18Ø I=1,M                                                      DRV 189Ø
        IF (W(I).EQ.Ø.Ø) KZ = KZ + 1                                    DRV 19ØØ
        WRITE (NW,9991Ø) (A(I,J),J=1,N),(B(I,K),K=1,L),W(I)             DRV 191Ø
  18Ø CONTINUE                                                          DRV 192Ø
C                                                                       DRV 193Ø
      GO TO (19Ø,2ØØ), MODE                                             DRV 194Ø
C                                                                       DRV 195Ø
  19Ø CALL L2A(M, N, M1, L, A, B, W, TOL, MM, NN,                       DRV 196Ø
     * N1, IPIVOT, X, RES, R, Q, C, IFAULT)                            DRV 197Ø
C                                                                       DRV 198Ø
      GO TO 21Ø                                                         DRV 199Ø
C                                                                       DRV 2ØØØ
  2ØØ CALL L2B(M, N, M1, L, A, B, W, TOL, MM, NN, MMPNN,                DRV 2Ø1Ø
     * N1, IPIVOT, X, RES, QR, C, IFAULT)                              DRV 2Ø2Ø
C                                                                       DRV 2Ø3Ø
C PRINT COMPUTED RESULTS.                                               DRV 2Ø4Ø
C                                                                       DRV 2Ø5Ø
  21Ø WRITE (NW,999ØØ)                                                  DRV 2Ø6Ø
      WRITE (NW,9989Ø) MODE,IFAULT                                      DRV 2Ø7Ø
      IF (IFAULT.GE.1 .AND. IFAULT.LE.4) GO TO 39Ø                      DRV 2Ø8Ø
      WRITE (NW,9988Ø) N1                                               DRV 2Ø9Ø
      IF (N1.EQ.Ø) GO TO 39Ø                                            DRV 21ØØ
      IF (N1.LT.M1) GO TO 39Ø                                           DRV 211Ø
      WRITE (NW,9987Ø)                                                  DRV 212Ø
      WRITE (NW,9986Ø) (IPIVOT(J),J=1,N1)                               DRV 213Ø
      IF (N1.EQ.N) GO TO 22Ø                                            DRV 214Ø
      N1P1 = N1 + 1                                                     DRV 215Ø
      WRITE (NW,9985Ø)                                                  DRV 216Ø
      WRITE (NW,9986Ø) (IPIVOT(J),J=N1P1,N)                             DRV 217Ø
  22Ø NDF = M - N1 - KZ                                                 DRV 218Ø
      WRITE (NW,9972Ø) KZ,NDF                                           DRV 219Ø
      WRITE (NW,9984Ø)                                                  DRV 22ØØ
      DO 24Ø K=1,L                                                      DRV 221Ø
        K2 = L + K                                                      DRV 222Ø
        IF (C(K).LT.Ø.Ø) GO TO 23Ø                                      DRV 223Ø
        FAIL(K) = .FALSE.                                               DRV 224Ø
        WRITE (NW,9983Ø) K,C(K),C(K2)                                   DRV 225Ø
        GO TO 24Ø                                                       DRV 226Ø
  23Ø   FAIL(K) = .TRUE.                                                DRV 227Ø
        C(K) = -C(K)                                                    DRV 228Ø
        WRITE (NW,9982Ø) K,C(K),C(K2)                                   DRV 229Ø
  24Ø CONTINUE                                                          DRV 23ØØ
      IF (IFAULT.EQ.7) GO TO 39Ø                                        DRV 231Ø
      DO 35Ø K=1,L                                                      DRV 232Ø
C                                                                       DRV 233Ø
C COMPUTE SUM OF SQUARED RESIDUALS, NORM OF RESIDUALS, RESIDUAL         DRV 234Ø
C STANDARD DEVIATION, STANDARD DEVIATIONS OF COEFFICIENTS, AND          DRV 235Ø
C PREDICTED VALUES.  PRINT THESE QUANTITIES, TOGETHER WITH              DRV 236Ø
C COEFFICIENTS, OBSERVED VALUES AND RESIDUALS.                          DRV 237Ø
C                                                                       DRV 238Ø
        IF (FAIL(K)) GO TO 35Ø                                          DRV 239Ø
        WRITE (NW,9981Ø) K                                              DRV 24ØØ
        SS = Ø.Ø                                                        DRV 241Ø
        IF (M.EQ.M1) GO TO 31Ø                                          DRV 242Ø
        M1P1 = M1 + 1                                                   DRV 243Ø
        DO 25Ø I=M1P1,M                                                 DRV 244Ø
          SS = SS + (RES(I,K)*W(I))**2                                  DRV 245Ø
  25Ø   CONTINUE                                                        DRV 246Ø
        IF (M.LE.N1+KZ) GO TO 31Ø                                       DRV 247Ø
        IF (MODE.EQ.2 .AND. N1.LT.N) GO TO 31Ø                          DRV 248Ø
        SD = SQRT(SS/FLOAT(NDF))                                        DRV 249Ø
        IF (MODE.EQ.2) GO TO 27Ø                                        DRV 25ØØ
        DO 26Ø J=1,N                                                    DRV 251¢
          IF (R(J,J).LT.Ø.Ø) R(J,J) = Ø.Ø                               DRV 252¢
          SDX(J) = SD*SQRT(R(J,J))                                      DRV 253Ø
  26Ø   CONTINUE                                                        DRV 254Ø
```

```
      GO TO 29Ø                                                          DRV 255Ø
 27Ø  DO 28Ø J=1,N                                                       DRV 256Ø
        IF (QR(J,J).LT.Ø.Ø) QR(J,J) = Ø.Ø                                DRV 257Ø
        SDX(J) = SD*SQRT(QR(J,J))                                        DRV 258Ø
 28Ø  CONTINUE                                                           DRV 259Ø
 29Ø  WRITE (NW,9980Ø)                                                   DRV 26ØØ
      DO 3ØØ J=1,N                                                       DRV 261Ø
        WRITE (NW,9977Ø) J,X(J,K),SDX(J)                                 DRV 262Ø
 3ØØ  CONTINUE                                                           DRV 263Ø
      GO TO 33Ø                                                          DRV 264Ø
 31Ø  WRITE (NW,9979Ø)                                                   DRV 265Ø
      DO 32Ø J=1,N                                                       DRV 266Ø
        WRITE (NW,9977Ø) J,X(J,K)                                        DRV 267Ø
 32Ø  CONTINUE                                                           DRV 268Ø
 33Ø  WRITE (NW,9978Ø)                                                   DRV 269Ø
      DO 34Ø I=1,M                                                       DRV 27ØØ
        Z = B(I,K) - RES(I,K)                                            DRV 271Ø
        WRITE (NW,9977Ø) I,B(I,K),Z,RES(I,K)                             DRV 272Ø
 34Ø  CONTINUE                                                           DRV 273Ø
      SNORM = SQRT(SS)                                                   DRV 274Ø
      WRITE (NW,9976Ø) SS,SNORM                                          DRV 275Ø
      IF ((MODE.EQ.2 .AND. N1.LT.N) .OR. (M.EQ.N1+KZ)) GO TO 35Ø         DRV 276Ø
      WRITE (NW,9975Ø) SD                                                DRV 277Ø
 35Ø CONTINUE                                                            DRV 278Ø
C                                                                        DRV 279Ø
C PRINT LOWER TRIANGULAR PORTION OF SYMMETRIC UNSCALED COVARIANCE        DRV 28ØØ
C MATRIX.                                                                DRV 281Ø
C                                                                        DRV 282Ø
      IF (MODE.EQ.2 .AND. N1.LT.N) GO TO 39Ø                             DRV 283Ø
      WRITE (NW,9974Ø)                                                   DRV 284Ø
      IF (MODE.EQ.2) GO TO 37Ø                                           DRV 285Ø
      DO 36Ø I=1,N                                                       DRV 286Ø
        WRITE (NW,9991Ø) (R(I,J),J=1,I)                                  DRV 287Ø
 36Ø CONTINUE                                                            DRV 288Ø
      GO TO 39Ø                                                          DRV 289Ø
 37Ø DO 38Ø I=1,N                                                        DRV 29ØØ
        WRITE (NW,9991Ø) (QR(I,J),J=1,I)                                 DRV 291Ø
 38Ø CONTINUE                                                            DRV 292Ø
 39Ø READ (NR,9973Ø) IFDONE                                              DRV 293Ø
      IF (IFDONE.EQ.1) GO TO 1Ø                                          DRV 294Ø
C                                                                        DRV 295Ø
      STOP                                                               DRV 296Ø
C                                                                        DRV 297Ø
C FORMAT STATEMENTS.                                                     DRV 298Ø
C                                                                        DRV 299Ø
9999Ø FORMAT (8ØA1)                                                      DRV 3ØØØ
9998Ø FORMAT (1H1,115(1H*),4X,7HPROBLEM,I4)                              DRV 3Ø1Ø
9997Ø FORMAT (1HØ,8ØA1)                                                  DRV 3Ø2Ø
9996Ø FORMAT (6I5,5X,F1Ø.Ø)                                              DRV 3Ø3Ø
9995Ø FORMAT (1HØ,3X,1HM,4X,1HN,3X,2HM1,4X,1HL,5X,5HITYPE,5X,4HIWGH,     DRV 3Ø4Ø
     * 1HT,6X,4HMODE,7X,3HTOL)                                           DRV 3Ø5Ø
9994Ø FORMAT (4I5,3I1Ø,G15.8)                                            DRV 3Ø6Ø
9993Ø FORMAT (8HØFORMAT ,8ØA1)                                           DRV 3Ø7Ø
9992Ø FORMAT (41HØMATRIX A, MATRIX B AND VECTOR OF WEIGHTS/)             DRV 3Ø8Ø
9991Ø FORMAT (1X,8G15.8)                                                 DRV 3Ø9Ø
999ØØ FORMAT (17HØCOMPUTED RESULTS)                                      DRV 31ØØ
9989Ø FORMAT (7HØMODE =,I4,5X,8HIFAULT =,I4)                             DRV 311Ø
9988Ø FORMAT (44HØN1 = COMPUTED RANK OF SYSTEM OF EQUATIONS =,I4)        DRV 312Ø
9987Ø FORMAT (5ØHØCOLUMNS OF H = (SQRT(W))*A WERE SELECTED BY THE P,     DRV 313Ø
     * 37HIVOTING SCHEME IN THE FOLLOWING ORDER/)                        DRV 314Ø
9986Ø FORMAT (3ØI4)                                                      DRV 315Ø
9985Ø FORMAT (5ØHØTHE FOLLOWING COLUMNS OF H ARE LINEARLY DEPENDENT,     DRV 316Ø
     * 48H.  IF MODE 1, THEY DID NOT ENTER THE REGRESSION./6H IF MO,     DRV 317Ø
     * 24HDE 2, THEY ENTERED LAST./)                                     DRV 318Ø
9984Ø FORMAT (1HØ,15X,9HREPORT ON,12X,9HNUMBER OF,4X,11HESTIMATED N,     DRV 319Ø
     * 16HUMBER OF CORRECT/13H B-VECTOR NO.,3X,11HCONVERGENCE,1ØX,       DRV 32ØØ
     * 1ØHITERATIONS,3X,26HDIGITS IN INITIAL SOLUTION/)                  DRV 321Ø
9983Ø FORMAT (I7,9X,9HCONVERGED,14X,F4.Ø,1ØX,G15.8)                      DRV 322Ø
9982Ø FORMAT (I7,9X,18HFAILED TO CONVERGE,5X,F4.Ø,1ØX,G15.8)             DRV 323Ø
9981Ø FORMAT (26HØSOLUTION FOR B-VECTOR NO.,I3)                          DRV 324Ø
998ØØ FORMAT (1HØ,27X,18HSTANDARD DEVIATION/5X,1HJ,4X,1ØHCOEFFICIEN,     DRV 325Ø
     * 4HT(J),5X,17HOF COEFFICIENT(J)/)                                 DRV 326Ø
9979Ø FORMAT (1HØ,4X,1HJ,4X,14HCOEFFICIENT(J)/)                          DRV 327Ø
9978Ø FORMAT (1HØ,4X,1HI,4X,11HOBSERVED(I),9X,12HPREDICTED(I),1ØX,       DRV 328Ø
     * 11HRESIDUAL(I)/)                                                 DRV 329Ø
9977Ø FORMAT (I6,G17.8,2G21.8)                                           DRV 33ØØ
```

```
99760 FORMAT (30H0SUM OF SQUARED RESIDUALS    =,G15.8/1X,8HNORM OF ,    DRV 3310
     * 9HRESIDUALS,11X,1H=,G15.8)                                      DRV 3320
99750 FORMAT (30H RESIDUAL STANDARD DEVIATION =,G15.8)                  DRV 3330
99740 FORMAT (27H0UNSCALED COVARIANCE MATRIX/)                          DRV 3340
99730 FORMAT (I5)                                                       DRV 3350
99720 FORMAT(25H0NUMBER OF ZERO WEIGHTS =,I3,5X,17HDEG. OF FREEDOM =,I3)DRV 3360
      END                                                               DRV 3370


      SUBROUTINE L2A(M, N, M1, L, A, B, W, TOL, MM, NN,                 L2A   10
     * N1, IPIVOT, X, RES, R, Q, C, IFAULT)                            L2A   20
C ** PURPOSE **                                                         L2A   30
C SUBROUTINE L2A COMPUTES LEAST SQUARES SOLUTIONS TO OVERDETERMINED     L2A   40
C SYSTEMS OF LINEAR EQUATIONS.  THE METHOD USED IS A MODIFIED           L2A   50
C GRAM-SCHMIDT ORTHOGONAL DECOMPOSITION WITH ITERATIVE REFINEMENT OF    L2A   60
C THE SOLUTION.  THE SOLUTION MAY BE SUBJECT TO LINEAR EQUALITY         L2A   70
C CONSTRAINTS.  OUTPUT INCLUDES THE LEAST SQUARES COEFFICIENTS,         L2A   80
C RESIDUALS, UNSCALED COVARIANCE MATRIX, AND INFORMATION ON THE         L2A   90
C BEHAVIOR OF THE ITERATIVE REFINEMENT PROCEDURE.                       L2A  100
C MATRIX A IS THE GIVEN MATRIX OF A SYSTEM OF M LINEAR EQUATIONS IN N   L2A  110
C UNKNOWNS, AND MATRIX W IS A GIVEN DIAGONAL MATRIX OF WEIGHTS WITH ALL L2A  120
C DIAGONAL ELEMENTS NONNEGATIVE.  LET H = (SQRT(W))*A.                  L2A  130
C IN THE EVENT THAT N1 (THE COMPUTED RANK OF MATRIX H) IS LESS THAN N   L2A  140
C (THE NUMBER OF UNKNOWN COEFFICIENTS), THE ORIGINAL MATRIX H (M BY N)  L2A  150
C IS REPLACED BY A SMALLER MATRIX (M BY N1) WHOSE COLUMNS ARE LINEARLY  L2A  160
C INDEPENDENT, AND A SOLUTION IS SOUGHT FOR THE SMALLER SYSTEM OF       L2A  170
C EQUATIONS.  THUS N - N1 COLUMNS OF THE ORIGINAL MATRIX H ARE DELETED, L2A  180
C AND COEFFICIENTS CORRESPONDING TO THESE N - N1 DELETED COLUMNS WILL   L2A  190
C BE SET EQUAL TO ZERO.                                                 L2A  200
C                                                                       L2A  210
C ** INPUT VARIABLES **                                                 L2A  220
C M       TOTAL NUMBER OF EQUATIONS.                                    L2A  230
C N       NUMBER OF UNKNOWN COEFFICIENTS.                               L2A  240
C M1      NUMBER OF LINEAR CONSTRAINTS (0.LE.M1.LE.M AND M1.LE.N).      L2A  250
C L       NUMBER OF RIGHT-HAND SIDES (VECTORS OF OBSERVATIONS).         L2A  260
C A       TWO-DIMENSIONAL ARRAY OF SIZE (MM,N).  ON ENTRY, THE ARRAY A  L2A  270
C         CONTAINS THE GIVEN MATRIX OF A SYSTEM OF M LINEAR EQUATIONS   L2A  280
C         IN N UNKNOWNS, WHERE THE FIRST M1 EQUATIONS ARE TO BE         L2A  290
C         SATISFIED EXACTLY.  A IS LEFT INTACT ON EXIT.                 L2A  300
C B       TWO-DIMENSIONAL ARRAY OF SIZE (MM,L).  ON ENTRY, B CONTAINS   L2A  310
C         THE L GIVEN RIGHT-HAND SIDES (VECTORS OF OBSERVATIONS).  B IS L2A  320
C         LEFT INTACT ON EXIT.                                          L2A  330
C W       VECTOR OF SIZE M.  ON ENTRY, W CONTAINS THE DIAGONAL ELEMENTS L2A  340
C         OF A GIVEN DIAGONAL MATRIX OF WEIGHTS, ALL NONNEGATIVE.       L2A  350
C         (THE FIRST M1 ELEMENTS OF W ARE SET EQUAL TO 1.0 BY THE       L2A  360
C         PROGRAM WHEN M1 IS GREATER THAN ZERO.)  ON EXIT, THE ORIGINAL L2A  370
C         ELEMENTS OF W HAVE BEEN REPLACED BY THEIR SQUARE ROOTS.       L2A  380
C TOL     PARAMETER USED IN DETERMINING THE RANK OF MATRIX H.           L2A  390
C         NOTE --                                                       L2A  400
C         (1) IF TOL EQUALS ZERO, THE TOLERANCE USED IN SUBROUTINE      L2A  410
C             DECOM1 WILL BE BASED ON MACHINE PRECISION.                L2A  420
C         (2) IF TOL IS GREATER THAN ZERO, THIS VALUE OF TOL WILL BE    L2A  430
C             USED IN SETTING AN ABSOLUTE TOLERANCE FOR COMPARISON WITH L2A  440
C             DIAGONAL ELEMENTS OF THE TRIANGULAR MATRIX OBTAINED IN    L2A  450
C             SUBROUTINE DECOM1.  THE VALUE OF TOL CAN BE BASED ON      L2A  460
C             KNOWLEDGE CONCERNING THE ACCURACY OF THE DATA.            L2A  470
C MM      DIMENSIONING PARAMETER SPECIFYING MAXIMUM NUMBER OF ROWS IN   L2A  480
C         THE ARRAYS A, B, RES AND Q.  MM MUST SATISFY MM.GE.M.         L2A  490
C NN      DIMENSIONING PARAMETER SPECIFYING MAXIMUM NUMBER OF ROWS IN   L2A  500
C         THE ARRAYS X AND R.  NN MUST SATISFY NN.GE.N.                 L2A  510
C                                                                       L2A  520
C ** OUTPUT VARIABLES AND INTERNAL VARIABLES **                         L2A  530
C N1      COMPUTED RANK OF MATRIX H, WHERE H = (SQRT(W))*A.             L2A  540
C IPIVOT VECTOR OF SIZE N.  ON EXIT, THIS ARRAY RECORDS THE ORDER       L2A  550
C         IN WHICH THE COLUMNS OF H WERE SELECTED BY THE PIVOTING       L2A  560
C         SCHEME IN THE COURSE OF THE ORTHOGONAL DECOMPOSITION.         L2A  570
C         WHENEVER N1.LT.N, THE FIRST N1 ELEMENTS OF IPIVOT INDICATE    L2A  580
C         WHICH COLUMNS OF H WERE FOUND TO BE LINEARLY INDEPENDENT.     L2A  590
C X       TWO-DIMENSIONAL ARRAY OF SIZE (NN,L).  ON EXIT, X CONTAINS    L2A  600
C         THE SOLUTION VECTORS.                                         L2A  610
C RES     TWO-DIMENSIONAL ARRAY OF SIZE (MM,L).  ON EXIT, RES CONTAINS  L2A  620
C         THE RESIDUAL VECTORS.                                         L2A  630
C R       TWO-DIMENSIONAL ARRAY OF SIZE (NN,N).  ON EXIT, R CONTAINS    L2A  640
C         THE LOWER TRIANGULAR PORTION OF THE SYMMETRIC UNSCALED        L2A  650
C         COVARIANCE MATRIX.  (THIS ARRAY IS USED INTERNALLY TO STORE   L2A  660
```

```
C        RESULTS FROM SUBROUTINE DECOM1 WHICH ARE DESTROYED IN          L2A  67Ø
C        COMPUTING THE COVARIANCE MATRIX.)                              L2A  68Ø
C Q      TWO-DIMENSIONAL ARRAY OF SIZE (MM,N) USED INTERNALLY ONLY.     L2A  69Ø
C C      VECTOR HAVING AT LEAST 4*(M+N)+2*L ELEMENTS USED (1) FOR       L2A  7ØØ
C        INTERNAL WORK SPACE AND (2) FOR RETURNING INFORMATION ON THE   L2A  71Ø
C        BEHAVIOR OF THE ITERATIVE REFINEMENT PROCEDURE.                L2A  72Ø
C        (A) NUMIT IS THE NUMBER OF ITERATIONS CARRIED OUT DURING THE   L2A  73Ø
C            ITERATIVE REFINEMENT IN ATTEMPTING TO OBTAIN A SOLUTION    L2A  74Ø
C            FOR THE K-TH RIGHT-HAND SIDE.                              L2A  75Ø
C            ON EXIT, C(K) = +NUMIT IF THE SOLUTION CONVERGED, AND      L2A  76Ø
C                     C(K) = -NUMIT IF THE SOLUTION FAILED TO CONVERGE. L2A  77Ø
C        (B) DIGITX GIVES AN ESTIMATE OF THE NUMBER OF CORRECT DIGITS   L2A  78Ø
C            IN THE INITIAL SOLUTION OF THE COEFFICIENTS FOR THE K-TH   L2A  79Ø
C            RIGHT-HAND SIDE.  ON EXIT, C(K+L) = DIGITX.                L2A  8ØØ
C IFAULT FAULT INDICATOR WHICH IS ZERO IF NO ERRORS WERE ENCOUNTERED    L2A  81Ø
C        AND POSITIVE IF ERRORS WERE DETECTED OR IF EVIDENCE OF SEVERE  L2A  82Ø
C        ILL-CONDITIONING WAS FOUND.  DIAGNOSTIC MESSAGES ARE PRINTED   L2A  83Ø
C        FROM SUBROUTINE ERROR.  IF IFAULT IS SET EQUAL TO 1, 2, 3, 4,  L2A  84Ø
C        5, 6 OR 7, EXECUTION IS TERMINATED.  EXECUTION CONTINUES WHEN  L2A  85Ø
C        IFAULT IS SET EQUAL TO 8, 9 OR 1Ø PROVIDED THAT A SOLUTION     L2A  86Ø
C        WAS OBTAINED FOR AT LEAST ONE RIGHT-HAND SIDE.  THE VALUE OF   L2A  87Ø
C        IFAULT IS USED TO INDICATE THE FOLLOWING --                    L2A  88Ø
C        Ø = NO ERRORS ENCOUNTERED.                                     L2A  89Ø
C        1 = BAD INPUT PARAMETER (M, N OR L).                           L2A  9ØØ
C        2 = BAD INPUT PARAMETER (M1).                                  L2A  91Ø
C        3 = BAD DIMENSION.  EITHER M.GT.MM OR N.GT.NN.                 L2A  92Ø
C        4 = AT LEAST ONE WEIGHT IS NEGATIVE.                           L2A  93Ø
C        5 = EITHER MATRIX H OR MATRIX OF CONSTRAINTS EQUALS ZERO.      L2A  94Ø
C        6 = CONSTRAINTS ARE LINEARLY DEPENDENT.                        L2A  95Ø
C        7 = ALL SOLUTIONS FAILED TO CONVERGE.                          L2A  96Ø
C        8 = SOLUTION FAILED TO CONVERGE FOR AT LEAST ONE RIGHT-HAND    L2A  97Ø
C            SIDE.                                                      L2A  98Ø
C        9 = LARGE NUMBER OF ITERATIONS REQUIRED FOR CONVERGENCE.       L2A  99Ø
C       1Ø = ESTIMATED NUMBER OF DIGITS IN INITIAL SOLUTION OF          L2A 1ØØØ
C            COEFFICIENTS IS SMALL.                                     L2A 1Ø1Ø
C       11 = DIAGONAL ELEMENT OF COVARIANCE MATRIX WAS COMPUTED TO BE   L2A 1Ø2Ø
C            NEGATIVE OWING TO ROUNDING ERROR.                          L2A 1Ø3Ø
C                                                                       L2A 1Ø4Ø
C ** SUBROUTINES REQUIRED **                                           L2A 1Ø5Ø
C SUBROUTINE DECOM1                                                    L2A 1Ø6Ø
C        USES MODIFIED GRAM-SCHMIDT ALGORITHM WITH PIVOTING TO          L2A 1Ø7Ø
C        OBTAIN AN ORTHOGONAL DECOMPOSITION OF THE INPUT MATRIX.        L2A 1Ø8Ø
C SUBROUTINE SOLVE1                                                    L2A 1Ø9Ø
C        COMPUTES COEFFICIENTS AND RESIDUALS.  ITERATIVE REFINEMENT IS  L2A 11ØØ
C        USED TO IMPROVE THE ACCURACY OF THE INITIAL SOLUTION.          L2A 111Ø
C SUBROUTINE COVAR                                                     L2A 112Ø
C        COMPUTES UNSCALED COVARIANCE MATRIX OF THE COEFFICIENTS.       L2A 113Ø
C SUBROUTINE ERROR                                                     L2A 114Ø
C        PRINTS ERROR DIAGNOSTICS WHEN ERRORS ARE DETECTED OR WHEN      L2A 115Ø
C        EVIDENCE OF SEVERE ILL-CONDITIONING IS FOUND.                  L2A 116Ø
C                                                                       L2A 117Ø
C ** STORAGE REQUIREMENTS **                                           L2A 118Ø
C THE STORAGE REQUIRED FOR THE DIMENSIONED ARRAYS IN SUBROUTINE L2A IS  L2A 119Ø
C   M*(2*N + 2*L + 5) + N*(N + L + 5) + 2*L                            L2A 12ØØ
C LOCATIONS.  ALL ARRAYS REQUIRED IN SUBROUTINES CALLED BY L2A ARE     L2A 121Ø
C DECLARED HEREIN AND ARE TRANSMITTED ONLY THROUGH PARAMETER LISTS OF  L2A 122Ø
C CALL-SEQUENCES.                                                      L2A 123Ø
C                                                                       L2A 124Ø
C ** PRECISION OF ARITHMETIC CALCULATIONS **                           L2A 125Ø
C SINGLE PRECISION ARITHMETIC IS USED FOR ALL CALCULATIONS EXCEPT THE  L2A 126Ø
C DOUBLE PRECISION ACCUMULATION OF INNER PRODUCTS.  (THE VARIABLE SUM   L2A 127Ø
C IS DECLARED TO BE DOUBLE PRECISION IN SUBROUTINES DECOM1, SOLVE1 AND L2A 128Ø
C COVAR.)  IT IS ESSENTIAL FOR THE SUCCESS OF THE ITERATIVE REFINEMENT  L2A 129Ø
C PROCEDURE THAT INNER PRODUCTS BE ACCUMULATED IN DOUBLE PRECISION.     L2A 13ØØ
C                                                                       L2A 131Ø
C ** CONVERSION OF THE PROGRAM TO DOUBLE PRECISION **                  L2A 132Ø
C ********************************************************************* L2A 133Ø
C * ON COMPUTERS HAVING SHORT WORD LENGTH (AS THE IBM 36Ø/37Ø) IT MAY * L2A 134Ø
C * BE DESIRABLE TO PERFORM ALL CALCULATIONS IN DOUBLE PRECISION.  IN * L2A 135Ø
C * THIS CASE, THE ITERATIVE REFINEMENT PRESENTLY INCLUDED IN SOLVE1  * L2A 136Ø
C * SHOULD BE OMITTED.                                               * L2A 137Ø
C * TO CONVERT THE PROGRAM TO DOUBLE PRECISION, THE FOLLOWING        * L2A 138Ø
C * APPROACH IS SUGGESTED.                                           * L2A 139Ø
C *                                                                  * L2A 14ØØ
C * 1. VARIABLES PRESENTLY DECLARED TO BE REAL SHOULD BE DECLARED    * L2A 141Ø
```

```
C *    DOUBLE PRECISION.  THOSE TYPED INTEGER, DOUBLE PRECISION AND   * L2A 142Ø
C *    LOGICAL SHOULD NOT BE CHANGED.                                 * L2A 143Ø
C * 2. THE USE OF FAIL, NUMIT AND DIGITX SHOULD BE OMITTED.           * L2A 144Ø
C * 3. DESCRIPTION OF VARIABLE C (AT L2A 7ØØ-8ØØ) SHOULD READ --     * L2A 145Ø
C *    C  VECTOR HAVING AT LEAST 4*(M+N) ELEMENTS USED ONLY FOR       * L2A 146Ø
C *       INTERNAL WORK SPACE.                                        * L2A 147Ø
C * 4. THE VALUE OF ETA (AT L2A 193Ø) SHOULD BE SET SO THAT IT IS THE * L2A 148Ø
C *    SMALLEST POSITIVE DOUBLE PRECISION NUMBER SUCH THAT 1.Ø + ETA  * L2A 149Ø
C *    IS GREATER THAN 1.Ø IN DOUBLE PRECISION ARITHMETIC.            * L2A 15ØØ
C *    FOR IBM COMPUTER TYPE, ETA = 16.**(-13)                        * L2A 151Ø
C *    FOR UNIVAC COMPUTER TYPE, ETA = 2.**(-59)                      * L2A 152Ø
C * 5. THE FOLLOWING FORTRAN FUNCTIONS SHOULD BE CHANGED --           * L2A 153Ø
C *      SINGLE PRECISION NAME     DOUBLE PRECISION NAME               * L2A 154Ø
C *            DBLE(X)                X                               * L2A 155Ø
C *            FLOAT(N)               DBLE(FLOAT(N))                  * L2A 156Ø
C *            SQRT(X)                DSQRT(X)                        * L2A 157Ø
C *    DBLE(X) IS USED IN SUBROUTINES DECOM1, SOLVE1 AND COVAR.       * L2A 158Ø
C *    FLOAT(N) IS USED IN SUBROUTINE DECOM1.                         * L2A 159Ø
C *    SQRT(X) IS USED IN SUBROUTINE L2A.                             * L2A 16ØØ
C * 6. IT MAY BE NECESSARY OR DESIRABLE TO CHANGE CERTAIN FORMATS IN  * L2A 161Ø
C *    SUBROUTINE ERROR, REPLACING G SPECIFICATIONS BY D.             * L2A 162Ø
C * 7. REPLACE STATEMENT L2A 245Ø BY A STATEMENT READING              * L2A 163Ø
C *          K3 = 1                                                   * L2A 164Ø
C * 8. FURTHER DETAILS ARE GIVEN IN SUBROUTINE SOLVE1 IN CONNECTION   * L2A 165Ø
C *    WITH THE OMISSION OF ITERATIVE REFINEMENT.                     * L2A 166Ø
C * 9. IN SUBROUTINE L2A, STATEMENTS L2A 96Ø-1Ø1Ø, 179Ø-18ØØ, 199Ø,   * L2A 167Ø
C *    232Ø-233Ø, 243Ø-244Ø, 297Ø, 317Ø-346Ø AND 348Ø-351Ø SHOULD BE  * L2A 168Ø
C *    OMITTED.                                                       * L2A 169Ø
C *    STATEMENT NUMBERS GIVEN HERE REFER TO THOSE IN THE RIGHT-HAND  * L2A 17ØØ
C *    MARGIN.  CERTAIN COMMENTS IN SUBROUTINE L2A DO NOT APPLY TO    * L2A 171Ø
C *    THE DOUBLE PRECISION VERSION.                                  * L2A 172Ø
C *                                                                   * L2A 173Ø
C ********************************************************************* L2A 174Ø
C                                                                       L2A 175Ø
      INTEGER IPIVOT(N)                                                 L2A 176Ø
      REAL A(MM,N), B(MM,L), C(1), ETA, Q(MM,N), R(NN,N),               L2A 177Ø
     * RES(MM,L), TOL, W(M), X(NN,L), Z                                 L2A 178Ø
      REAL DIGITX                                                       L2A 179Ø
      LOGICAL FAIL                                                      L2A 18ØØ
      LOGICAL SING                                                      L2A 181Ø
C                                                                       L2A 182Ø
C SET VALUE OF ETA, A MACHINE-DEPENDENT PARAMETER.                      L2A 183Ø
C ETA, THE RELATIVE MACHINE PRECISION, IS THE SMALLEST POSITIVE REAL    L2A 184Ø
C NUMBER SUCH THAT 1.Ø + ETA IS GREATER THAN 1.Ø IN FLOATING-POINT      L2A 185Ø
C ARITHMETIC.                                                           L2A 186Ø
C                                                                       L2A 187Ø
C FOR IBM COMPUTER TYPE, ETA = 16.**(-5)                                L2A 188Ø
C FOR UNIVAC COMPUTER TYPE, ETA = 2.**(-26)                             L2A 189Ø
C FOR CDC COMPUTER TYPE, ETA = 2.**(-47)                                L2A 19ØØ
C FOR HONEYWELL COMPUTER TYPE, ETA = 2.**(-27)                          L2A 191Ø
C                                                                       L2A 192Ø
      ETA = 2.**(-26)                                                   L2A 193Ø
C                                                                       L2A 194Ø
C DEFAULT VALUE FOR TOL IS ZERO.                                        L2A 195Ø
C                                                                       L2A 196Ø
      IF (TOL.LT.Ø.Ø) TOL = Ø.Ø                                         L2A 197Ø
      IFAULT = Ø                                                        L2A 198Ø
      KSUM = Ø                                                          L2A 199Ø
C                                                                       L2A 2ØØØ
C PERFORM INITIAL CHECKING OF INPUT PARAMETERS, DIMENSIONS AND          L2A 2Ø1Ø
C WEIGHTS FOR POSSIBLE ERRORS.                                          L2A 2Ø2Ø
C                                                                       L2A 2Ø3Ø
      IF (M.GT.Ø .AND. N.GT.Ø .AND. L.GT.Ø) GO TO 1Ø                    L2A 2Ø4Ø
      IFAULT = 1                                                        L2A 2Ø5Ø
      CALL ERROR(IFAULT, K, Z)                                          L2A 2Ø6Ø
      RETURN                                                            L2A 2Ø7Ø
   1Ø IF (M1.LE.M .AND. M1.LE.N .AND. M1.GE.Ø) GO TO 2Ø                 L2A 2Ø8Ø
      IFAULT = 2                                                        L2A 2Ø9Ø
      CALL ERROR(IFAULT, K, Z)                                          L2A 21ØØ
      RETURN                                                            L2A 211Ø
   2Ø IF (M.LE.MM .AND. N.LE.NN) GO TO 3Ø                               L2A 212Ø
      IFAULT = 3                                                        L2A 213Ø
      K = 1                                                             L2A 214Ø
      CALL ERROR(IFAULT, K, Z)                                          L2A 215Ø
      RETURN                                                            L2A 216Ø
   3Ø DO 4Ø I=1,M                                                       L2A 217Ø
```

```
      IF (M1.GT.Ø .AND. I.LE.M1) W(I) = 1.Ø                          L2A 218Ø
      IF (W(I).GE.Ø.Ø) GO TO 4Ø                                      L2A 219Ø
      IFAULT = 4                                                     L2A 22ØØ
      Z = W(I)                                                       L2A 221Ø
      CALL ERROR(IFAULT, I, Z)                                       L2A 222Ø
   4Ø CONTINUE                                                       L2A 223Ø
      IF (IFAULT.EQ.4) RETURN                                        L2A 224Ø
      DO 5Ø I=1,M                                                    L2A 225Ø
        W(I) = SQRT(W(I))                                            L2A 226Ø
   5Ø CONTINUE                                                       L2A 227Ø
C                                                                    L2A 228Ø
C SET PARAMETERS WHICH ALLOCATE VECTOR C TO CONTAIN CERTAIN FINAL    L2A 229Ø
C RESULTS AND ALSO TO BE USED AS WORK SPACE.                         L2A 23ØØ
C                                                                    L2A 231Ø
C K1 IS STARTING POINT FOR NUMIT AND FAIL, OF LENGTH L.              L2A 232Ø
C K2 IS STARTING POINT FOR DIGITX, OF LENGTH L.                      L2A 233Ø
C K3 IS STARTING POINT FOR D, OF LENGTH N.                           L2A 234Ø
C K4 IS STARTING POINT FOR K-TH COLUMN OF B, OF LENGTH M.            L2A 235Ø
C K5 IS STARTING POINT FOR K-TH COLUMN OF X, OF LENGTH N.            L2A 236Ø
C K6 IS STARTING POINT FOR K-TH COLUMN OF RES, OF LENGTH M.          L2A 237Ø
C K7 IS STARTING POINT FOR WORK SPACE OF LENGTH M.                   L2A 238Ø
C K8 IS STARTING POINT FOR WORK SPACE OF LENGTH M.                   L2A 239Ø
C K9 IS STARTING POINT FOR WORK SPACE OF LENGTH N.                   L2A 24ØØ
C K1Ø IS STARTING POINT FOR WORK SPACE OF LENGTH N.                  L2A 241Ø
C                                                                    L2A 242Ø
      K1 = 1                                                         L2A 243Ø
      K2 = K1 + L                                                    L2A 244Ø
      K3 = K2 + L                                                    L2A 245Ø
      K4 = K3 + N                                                    L2A 246Ø
      K5 = K4 + M                                                    L2A 247Ø
      K6 = K5 + N                                                    L2A 248Ø
      K7 = K6 + M                                                    L2A 249Ø
      K8 = K7 + M                                                    L2A 25ØØ
      K9 = K8 + M                                                    L2A 251Ø
      K1Ø = K9 + N                                                   L2A 252Ø
      K = K1Ø + N - 1                                                L2A 253Ø
C                                                                    L2A 254Ø
C MULTIPLY EACH ROW OF MATRIX A (M BY N) BY ITS APPROPRIATE WEIGHT AND L2A 255Ø
C STORE THE RESULT IN ARRAY Q.  SET ARRAYS C AND R EQUAL TO ZERO.    L2A 256Ø
C                                                                    L2A 257Ø
      DO 6Ø I=1,K                                                    L2A 258Ø
        C(I) = Ø.Ø                                                   L2A 259Ø
   6Ø CONTINUE                                                       L2A 26ØØ
      DO 9Ø J=1,N                                                    L2A 261Ø
        DO 7Ø I=1,M                                                  L2A 262Ø
          Q(I,J) = A(I,J)*W(I)                                       L2A 263Ø
   7Ø   CONTINUE                                                     L2A 264Ø
        DO 8Ø I=1,N                                                  L2A 265Ø
          R(I,J) = Ø.Ø                                               L2A 266Ø
   8Ø   CONTINUE                                                     L2A 267Ø
   9Ø CONTINUE                                                       L2A 268Ø
C                                                                    L2A 269Ø
C OBTAIN AN ORTHOGONAL DECOMPOSITION OF THE MATRIX Q AND COMPUTE ITS L2A 27ØØ
C RANK.                                                              L2A 271Ø
C                                                                    L2A 272Ø
      CALL DECOM1(M, N, M1, ETA, TOL, Q, R, C(K3), N1, IPIVOT, SING, L2A 273Ø
     * MM, NN)                                                       L2A 274Ø
C                                                                    L2A 275Ø
      IF (.NOT.SING) GO TO 11Ø                                       L2A 276Ø
      IF (N1.GT.Ø) GO TO 1ØØ                                         L2A 277Ø
      IFAULT = 5                                                     L2A 278Ø
      CALL ERROR(IFAULT, K, Z)                                       L2A 279Ø
      RETURN                                                         L2A 28ØØ
  1ØØ IFAULT = 6                                                     L2A 281Ø
      CALL ERROR(IFAULT, K, Z)                                       L2A 282Ø
      RETURN                                                         L2A 283Ø
C                                                                    L2A 284Ø
C SEEK A SOLUTION (COEFFICIENTS AND RESIDUALS) FOR EACH OF THE L LEAST L2A 285Ø
C SQUARES PROBLEMS WHOSE RIGHT-HAND SIDES ARE GIVEN IN THE ARRAY B.  L2A 286Ø
C                                                                    L2A 287Ø
  11Ø DO 2ØØ K=1,L                                                   L2A 288Ø
C K-TH RIGHT-HAND SIDE.                                              L2A 289Ø
        KØ = K4 - 1                                                  L2A 29ØØ
        DO 12Ø I=1,M                                                 L2A 291Ø
          KØ = KØ + 1                                                L2A 292Ø
          C(KØ) = B(I,K)                                             L2A 293Ø
```

```
  12Ø   CONTINUE                                                        L2A 294Ø
C                                                                       L2A 295Ø
       CALL SOLVE1(M, N, M1, A, C(K4), W, N1, IPIVOT, Q, R, C(K3),      L2A 296Ø
     *   ETA, FAIL, NUMIT, DIGITX,                                      L2A 297Ø
     *   C(K5), C(K6), C(K7), C(K8), C(K9), C(K1Ø), MM, NN)             L2A 298Ø
C                                                                       L2A 299Ø
       KØ = K5 - 1                                                      L2A 3ØØØ
       DO 13Ø J=1,N                                                     L2A 3Ø1Ø
         KØ = KØ + 1                                                    L2A 3Ø2Ø
         X(J,K) = C(KØ)                                                 L2A 3Ø3Ø
  13Ø  CONTINUE                                                         L2A 3Ø4Ø
       IF (M1.EQ.Ø) GO TO 15Ø                                           L2A 3Ø5Ø
       DO 14Ø I=1,M1                                                    L2A 3Ø6Ø
         RES(I,K) = Ø.Ø                                                 L2A 3Ø7Ø
  14Ø  CONTINUE                                                         L2A 3Ø8Ø
  15Ø  M1P1 = M1 + 1                                                    L2A 3Ø9Ø
       IF (M1P1.GT.M) GO TO 17Ø                                         L2A 31ØØ
       KØ = K6 + M1 - 1                                                 L2A 311Ø
       DO 16Ø I=M1P1,M                                                  L2A 312Ø
         KØ = KØ + 1                                                    L2A 313Ø
         RES(I,K) = C(KØ)                                               L2A 314Ø
  16Ø  CONTINUE                                                         L2A 315Ø
  17Ø  CONTINUE                                                         L2A 316Ø
C                                                                       L2A 317Ø
C FOR RIGHT-HAND SIDES WHERE CONVERGENCE OF A SOLUTION IS REPORTED,     L2A 318Ø
C A CHECK IS MADE FOR EVIDENCE OF SEVERE ILL-CONDITIONING.  SUCH        L2A 319Ø
C EVIDENCE IS FURNISHED BY LARGE VALUES OF NUMIT (NUMBER OF ITERATIONS  L2A 32ØØ
C BEFORE CONVERGENCE WAS OBTAINED) AND SMALL VALUES OF DIGITX           L2A 321Ø
C (ESTIMATE OF THE NUMBER OF CORRECT DIGITS IN THE INITIAL SOLUTION     L2A 322Ø
C OF THE COEFFICIENTS).  IF NUMIT EXCEEDS -ALOG1Ø(ETA) A DIAGNOSTIC    L2A 323Ø
C MESSAGE IS PRINTED TO WARN OF ILL-CONDITIONING.  IF DIGITX IS LESS    L2A 324Ø
C THAN Ø.5 (HALF A DECIMAL DIGIT) A SIMILAR DIAGNOSTIC MESSAGE IS       L2A 325Ø
C PRINTED.                                                              L2A 326Ø
C                                                                       L2A 327Ø
       C(K) = FLOAT(NUMIT)                                              L2A 328Ø
       IF (FAIL) C(K) = -C(K)                                           L2A 329Ø
       KØ = K2 + K - 1                                                  L2A 33ØØ
       C(KØ) = DIGITX                                                   L2A 331Ø
       IF (.NOT.FAIL) GO TO 18Ø                                         L2A 332Ø
C KSUM IS A TALLY OF SOLUTIONS WHICH FAILED TO CONVERGE.                L2A 333Ø
       KSUM = KSUM + 1                                                  L2A 334Ø
       IFAULT = 8                                                       L2A 335Ø
       CALL ERROR(IFAULT, K, Z)                                         L2A 336Ø
       GO TO 2ØØ                                                        L2A 337Ø
  18Ø  Z = -ALOG1Ø(ETA)                                                 L2A 338Ø
       IF (FLOAT(NUMIT).LE.Z) GO TO 19Ø                                 L2A 339Ø
       IFAULT = 9                                                       L2A 34ØØ
       Z = FLOAT(NUMIT)                                                 L2A 341Ø
       CALL ERROR(IFAULT, K, Z)                                         L2A 342Ø
  19Ø  IF (DIGITX.GE.Ø.5) GO TO 2ØØ                                     L2A 343Ø
       IFAULT = 1Ø                                                      L2A 344Ø
       Z = DIGITX                                                       L2A 345Ø
       CALL ERROR(IFAULT, K, Z)                                         L2A 346Ø
  2ØØ CONTINUE                                                          L2A 347Ø
      IF (KSUM.LT.L) GO TO 21Ø                                          L2A 348Ø
      IFAULT = 7                                                        L2A 349Ø
      CALL ERROR(IFAULT, K, Z)                                          L2A 35ØØ
      RETURN                                                            L2A 351Ø
C                                                                       L2A 352Ø
C COMPUTE THE UNSCALED COVARIANCE MATRIX OF THE COEFFICIENTS.           L2A 353Ø
C                                                                       L2A 354Ø
  21Ø CALL COVAR(N, M1, N1, IPIVOT, R, C(K3), C(K9), NN)                L2A 355Ø
C                                                                       L2A 356Ø
C IN CERTAIN PROBLEMS, SOME DIAGONAL TERMS OF THE UNSCALED COVARIANCE   L2A 357Ø
C MATRIX ARE EQUAL TO ZERO OR TO SMALL POSITIVE NUMBERS.  BECAUSE OF    L2A 358Ø
C ROUNDING ERRORS, COMPUTED VALUES FOR THESE TERMS MAY BE SMALL         L2A 359Ø
C NEGATIVE NUMBERS.  AN ERROR DIAGNOSTIC IS PRINTED IF ANY DIAGONAL     L2A 36ØØ
C TERM IS NEGATIVE.                                                     L2A 361Ø
C                                                                       L2A 362Ø
      DO 22Ø J=1,N                                                      L2A 363Ø
        IF (R(J,J).GE.Ø.Ø) GO TO 22Ø                                    L2A 364Ø
        IFAULT = 11                                                     L2A 365Ø
        Z = R(J,J)                                                      L2A 366Ø
        CALL ERROR(IFAULT, J, Z)                                        L2A 367Ø
  22Ø CONTINUE                                                          L2A 368Ø
      RETURN                                                            L2A 369Ø
      END                                                               L2A 37ØØ
```

```
      SUBROUTINE L2B(M, N, M1, L, A, B, W, TOL, MM, NN, MMPNN,          L2B   1Ø
     * N1, IPIVOT, X, RES, QR, C, IFAULT)                               L2B   2Ø
C ** PURPOSE **                                                         L2B   3Ø
C SUBROUTINE L2B COMPUTES LEAST SQUARES SOLUTIONS TO OVERDETERMINED     L2B   4Ø
C AND UNDERDETERMINED SYSTEMS OF LINEAR EQUATIONS.  THE METHOD USED IS  L2B   5Ø
C A MODIFIED GRAM-SCHMIDT ORTHOGONAL DECOMPOSITION WITH ITERATIVE       L2B   6Ø
C REFINEMENT OF THE SOLUTION.  THE SOLUTION MAY BE SUBJECT TO LINEAR    L2B   7Ø
C EQUALITY CONSTRAINTS.  OUTPUT INCLUDES THE LEAST SQUARES              L2B   8Ø
C COEFFICIENTS, RESIDUALS, UNSCALED COVARIANCE MATRIX, AND INFORMATION  L2B   9Ø
C ON THE BEHAVIOR OF THE ITERATIVE REFINEMENT PROCEDURE.                L2B  1ØØ
C MATRIX A IS THE GIVEN MATRIX OF A SYSTEM OF M LINEAR EQUATIONS IN N   L2B  11Ø
C UNKNOWNS, AND MATRIX W IS A GIVEN DIAGONAL MATRIX OF WEIGHTS WITH ALL L2B  12Ø
C DIAGONAL ELEMENTS NONNEGATIVE.  LET H = (SQRT(W))*A.                  L2B  13Ø
C IN THE EVENT THAT N1 (THE COMPUTED RANK OF MATRIX H) IS LESS THAN N   L2B  14Ø
C (THE NUMBER OF UNKNOWN COEFFICIENTS), A UNIQUE SOLUTION VECTOR HAVING L2B  15Ø
C N ELEMENTS CAN BE OBTAINED BY IMPOSING THE CONDITION THAT THE         L2B  16Ø
C SOLUTION BE OF MINIMAL EUCLIDEAN NORM.  SUCH A SOLUTION IS SOUGHT IN  L2B  17Ø
C THE CASE OF UNDERDETERMINED OR RANK-DEFICIENT PROBLEMS.               L2B  18Ø
C                                                                       L2B  19Ø
C ** INPUT VARIABLES **                                                 L2B  2ØØ
C M       TOTAL NUMBER OF EQUATIONS.                                    L2B  21Ø
C N       NUMBER OF UNKNOWN COEFFICIENTS.                               L2B  22Ø
C M1      NUMBER OF LINEAR CONSTRAINTS (Ø.LE.M1.LE.M AND M1.LE.N).      L2B  23Ø
C L       NUMBER OF RIGHT-HAND SIDES (VECTORS OF OBSERVATIONS).         L2B  24Ø
C A       TWO-DIMENSIONAL ARRAY OF SIZE (MM,N).  ON ENTRY, THE ARRAY A  L2B  25Ø
C         CONTAINS THE GIVEN MATRIX OF A SYSTEM OF M LINEAR EQUATIONS   L2B  26Ø
C         IN N UNKNOWNS, WHERE THE FIRST M1 EQUATIONS ARE TO BE         L2B  27Ø
C         SATISFIED EXACTLY.  A IS LEFT INTACT ON EXIT.                 L2B  28Ø
C B       TWO-DIMENSIONAL ARRAY OF SIZE (MM,L).  ON ENTRY, B CONTAINS   L2B  29Ø
C         THE L GIVEN RIGHT-HAND SIDES (VECTORS OF OBSERVATIONS).  B IS L2B  3ØØ
C         LEFT INTACT ON EXIT.                                          L2B  31Ø
C W       VECTOR OF SIZE M.  ON ENTRY, W CONTAINS THE DIAGONAL ELEMENTS L2B  32Ø
C         OF A GIVEN DIAGONAL MATRIX OF WEIGHTS, ALL NONNEGATIVE.       L2B  33Ø
C         (THE FIRST M1 ELEMENTS OF W ARE SET EQUAL TO 1.Ø BY THE       L2B  34Ø
C         PROGRAM WHEN M1 IS GREATER THAN ZERO.)  ON EXIT, THE ORIGINAL L2B  35Ø
C         ELEMENTS OF W HAVE BEEN REPLACED BY THEIR SQUARE ROOTS.       L2B  36Ø
C TOL     PARAMETER USED IN DETERMINING THE RANK OF MATRIX H.           L2B  37Ø
C         NOTE --                                                       L2B  38Ø
C         (1) IF TOL EQUALS ZERO, THE TOLERANCE USED IN SUBROUTINE      L2B  39Ø
C             DECOM2 WILL BE BASED ON MACHINE PRECISION.                L2B  4ØØ
C         (2) IF TOL IS GREATER THAN ZERO, THIS VALUE OF TOL WILL BE    L2B  41Ø
C             USED IN SETTING AN ABSOLUTE TOLERANCE FOR COMPARISON WITH L2B  42Ø
C             DIAGONAL ELEMENTS OF THE TRIANGULAR MATRIX OBTAINED IN    L2B  43Ø
C             SUBROUTINE DECOM2.  THE VALUE OF TOL CAN BE BASED ON      L2B  44Ø
C             KNOWLEDGE CONCERNING THE ACCURACY OF THE DATA.            L2B  45Ø
C MM      DIMENSIONING PARAMETER SPECIFYING MAXIMUM NUMBER OF ROWS IN   L2B  46Ø
C         THE ARRAYS A, B AND RES.  MM MUST SATISFY MM.GE.M.            L2B  47Ø
C NN      DIMENSIONING PARAMETER SPECIFYING MAXIMUM NUMBER OF ROWS IN   L2B  48Ø
C         THE ARRAY X.  NN MUST SATISFY NN.GE.N.                        L2B  49Ø
C MMPNN   DIMENSIONING PARAMETER SPECIFYING MAXIMUM NUMBER OF ROWS IN   L2B  5ØØ
C         THE ARRAY QR.  MMPNN MUST SATISFY MMPNN.GE.M+N.               L2B  51Ø
C                                                                       L2B  52Ø
C ** OUTPUT VARIABLES AND INTERNAL VARIABLES **                         L2B  53Ø
C N1      COMPUTED RANK OF MATRIX H, WHERE H = (SQRT(W))*A.             L2B  54Ø
C IPIVOT VECTOR OF SIZE N.  ON EXIT, THIS ARRAY RECORDS THE ORDER       L2B  55Ø
C         IN WHICH THE COLUMNS OF H WERE SELECTED BY THE PIVOTING       L2B  56Ø
C         SCHEME IN THE COURSE OF THE ORTHOGONAL DECOMPOSITION.         L2B  57Ø
C         WHENEVER N1.LT.N, THE FIRST N1 ELEMENTS OF IPIVOT INDICATE    L2B  58Ø
C         WHICH COLUMNS OF H WERE FOUND TO BE LINEARLY INDEPENDENT.     L2B  59Ø
C X       TWO-DIMENSIONAL ARRAY OF SIZE (NN,L).  ON EXIT, X CONTAINS    L2B  6ØØ
C         THE SOLUTION VECTORS.                                         L2B  61Ø
C RES     TWO-DIMENSIONAL ARRAY OF SIZE (MM,L).  ON EXIT, RES CONTAINS  L2B  62Ø
C         THE RESIDUAL VECTORS.                                         L2B  63Ø
C QR      TWO-DIMENSIONAL ARRAY OF SIZE (MMPNN,N).  ON EXIT, QR         L2B  64Ø
C         CONTAINS THE LOWER TRIANGULAR PORTION OF THE SYMMETRIC        L2B  65Ø
C         UNSCALED COVARIANCE MATRIX.  (THIS ARRAY IS USED INTERNALLY   L2B  66Ø
C         TO STORE RESULTS FROM SUBROUTINE DECOM2 WHICH ARE             L2B  67Ø
C         DESTROYED IN COMPUTING THE COVARIANCE MATRIX.)                L2B  68Ø
C C       VECTOR HAVING AT LEAST 6*(M+N)+2*L ELEMENTS USED (1) FOR      L2B  69Ø
C         INTERNAL WORK SPACE AND (2) FOR RETURNING INFORMATION ON THE  L2B  7ØØ
C         BEHAVIOR OF THE ITERATIVE REFINEMENT PROCEDURE.               L2B  71Ø
C         (A) NUMIT IS THE NUMBER OF ITERATIONS CARRIED OUT DURING THE  L2B  72Ø
C             ITERATIVE REFINEMENT IN ATTEMPTING TO OBTAIN A SOLUTION   L2B  73Ø
C             FOR THE K-TH RIGHT-HAND SIDE.                             L2B  74Ø
C             ON EXIT, C(K) = +NUMIT IF THE SOLUTION CONVERGED, AND     L2B  75Ø
C                      C(K) = -NUMIT IF THE SOLUTION FAILED TO CONVERGE. L2B  76Ø
```

```
C         (B) DIGITX GIVES AN ESTIMATE OF THE NUMBER OF CORRECT DIGITS   L2B  77Ø
C             IN THE INITIAL SOLUTION OF THE COEFFICIENTS FOR THE K-TH   L2B  78Ø
C             RIGHT-HAND SIDE.  ON EXIT, C(K+L) = DIGITX.                L2B  79Ø
C IFAULT FAULT INDICATOR WHICH IS ZERO IF NO ERRORS WERE ENCOUNTERED     L2B  8ØØ
C        AND POSITIVE IF ERRORS WERE DETECTED OR IF EVIDENCE OF SEVERE   L2B  81Ø
C        ILL-CONDITIONING WAS FOUND.  DIAGNOSTIC MESSAGES ARE PRINTED    L2B  82Ø
C        FROM SUBROUTINE ERROR.  IF IFAULT IS SET EQUAL TO 1, 2, 3, 4,   L2B  83Ø
C        5, 6 OR 7, EXECUTION IS TERMINATED.  EXECUTION CONTINUES WHEN   L2B  84Ø
C        IFAULT IS SET EQUAL TO 8, 9 OR 1Ø PROVIDED THAT A SOLUTION      L2B  85Ø
C        WAS OBTAINED FOR AT LEAST ONE RIGHT-HAND SIDE.  THE VALUE OF    L2B  86Ø
C        IFAULT IS USED TO INDICATE THE FOLLOWING --                     L2B  87Ø
C        Ø = NO ERRORS ENCOUNTERED.                                      L2B  88Ø
C        1 = BAD INPUT PARAMETER (M, N OR L).                            L2B  89Ø
C        2 = BAD INPUT PARAMETER (M1).                                   L2B  9ØØ
C        3 = BAD DIMENSION.  EITHER M.GT.MM, N.GT.NN OR M+N.GT.MMPNN.    L2B  91Ø
C        4 = AT LEAST ONE WEIGHT IS NEGATIVE.                            L2B  92Ø
C        5 = EITHER MATRIX H OR MATRIX OF CONSTRAINTS EQUALS ZERO.       L2B  93Ø
C        6 = CONSTRAINTS ARE LINEARLY DEPENDENT.                         L2B  94Ø
C        7 = ALL SOLUTIONS FAILED TO CONVERGE.                           L2B  95Ø
C        8 = SOLUTION FAILED TO CONVERGE FOR AT LEAST ONE RIGHT-HAND     L2B  96Ø
C            SIDE.                                                       L2B  97Ø
C        9 = LARGE NUMBER OF ITERATIONS REQUIRED FOR CONVERGENCE.        L2B  98Ø
C       1Ø = ESTIMATED NUMBER OF DIGITS IN INITIAL SOLUTION OF           L2B  99Ø
C            COEFFICIENTS IS SMALL.                                      L2B 1ØØØ
C       11 = DIAGONAL ELEMENT OF COVARIANCE MATRIX WAS COMPUTED TO BE    L2B 1Ø1Ø
C            NEGATIVE OWING TO ROUNDING ERROR.                           L2B 1Ø2Ø
C                                                                        L2B 1Ø3Ø
C ** SUBROUTINES REQUIRED **                                            L2B 1Ø4Ø
C SUBROUTINE DECOM2                                                     L2B 1Ø5Ø
C        USES MODIFIED GRAM-SCHMIDT ALGORITHM WITH PIVOTING TO          L2B 1Ø6Ø
C        OBTAIN AN ORTHOGONAL DECOMPOSITION OF THE INPUT MATRIX.        L2B 1Ø7Ø
C SUBROUTINE SOLVE2                                                     L2B 1Ø8Ø
C        COMPUTES COEFFICIENTS AND RESIDUALS.  ITERATIVE REFINEMENT IS  L2B 1Ø9Ø
C        USED TO IMPROVE THE ACCURACY OF THE INITIAL SOLUTION.          L2B 11ØØ
C SUBROUTINE SOLVE3                                                     L2B 111Ø
C        CALLED ONLY BY SUBROUTINE SOLVE2.                              L2B 112Ø
C SUBROUTINE COVAR                                                      L2B 113Ø
C        COMPUTES UNSCALED COVARIANCE MATRIX OF THE COEFFICIENTS.       L2B 114Ø
C SUBROUTINE ERROR                                                      L2B 115Ø
C        PRINTS ERROR DIAGNOSTICS WHEN ERRORS ARE DETECTED OR WHEN      L2B 116Ø
C        EVIDENCE OF SEVERE ILL-CONDITIONING IS FOUND.                  L2B 117Ø
C                                                                       L2B 118Ø
C ** STORAGE REQUIREMENTS **                                            L2B 119Ø
C THE STORAGE REQUIRED FOR THE DIMENSIONED ARRAYS IN SUBROUTINE L2B IS  L2B 12ØØ
C   M*(2*N + 2*L + 7) + N*(N + L + 7) + 2*L                             L2B 121Ø
C LOCATIONS.  ALL ARRAYS REQUIRED IN SUBROUTINES CALLED BY L2B ARE      L2B 122Ø
C DECLARED HEREIN AND ARE TRANSMITTED ONLY THROUGH PARAMETER LISTS OF   L2B 123Ø
C CALL-SEQUENCES.                                                       L2B 124Ø
C                                                                       L2B 125Ø
C ** PRECISION OF ARITHMETIC CALCULATIONS **                            L2B 126Ø
C SINGLE PRECISION ARITHMETIC IS USED FOR ALL CALCULATIONS EXCEPT THE   L2B 127Ø
C DOUBLE PRECISION ACCUMULATION OF INNER PRODUCTS.  (THE VARIABLE SUM   L2B 128Ø
C IS DECLARED TO BE DOUBLE PRECISION IN SUBROUTINES DECOM2, SOLVE2,     L2B 129Ø
C SOLVE3 AND COVAR.)  IT IS ESSENTIAL FOR THE SUCCESS OF THE ITERATIVE  L2B 13ØØ
C REFINEMENT PROCEDURE THAT INNER PRODUCTS BE ACCUMULATED IN DOUBLE     L2B 131Ø
C PRECISION.                                                            L2B 132Ø
C                                                                       L2B 133Ø
C ** CONVERSION OF THE PROGRAM TO DOUBLE PRECISION **                   L2B 134Ø
C ********************************************************************* L2B 135Ø
C * ON COMPUTERS HAVING SHORT WORD LENGTH (AS THE IBM 36Ø/37Ø) IT MAY * L2B 136Ø
C * BE DESIRABLE TO PERFORM ALL CALCULATIONS IN DOUBLE PRECISION.  IN * L2B 137Ø
C * THIS CASE, THE ITERATIVE REFINEMENT PRESENTLY INCLUDED IN SOLVE2  * L2B 138Ø
C * SHOULD BE OMITTED.                                                * L2B 139Ø
C * TO CONVERT THE PROGRAM TO DOUBLE PRECISION, THE FOLLOWING         * L2B 14ØØ
C * APPROACH IS SUGGESTED.                                            * L2B 141Ø
C *                                                                   * L2B 142Ø
C * 1. VARIABLES PRESENTLY DECLARED TO BE REAL SHOULD BE DECLARED     * L2B 143Ø
C *    DOUBLE PRECISION.  THOSE TYPED INTEGER, DOUBLE PRECISION AND   * L2B 144Ø
C *    LOGICAL SHOULD NOT BE CHANGED.                                 * L2B 145Ø
C * 2. THE USE OF FAIL, NUMIT AND DIGITX SHOULD BE OMITTED.           * L2B 146Ø
C * 3. DESCRIPTION OF VARIABLE C (AT L2B 69Ø-79Ø) SHOULD READ --      * L2B 147Ø
C *    C  VECTOR HAVING AT LEAST 6*(M+N) ELEMENTS USED ONLY FOR       * L2B 148Ø
C *       INTERNAL WORK SPACE.                                        * L2B 149Ø
C * 4. THE VALUE OF ETA (AT L2B 196Ø) SHOULD BE SET SO THAT IT IS THE * L2B 15ØØ
C *    SMALLEST POSITIVE DOUBLE PRECISION NUMBER SUCH THAT 1.Ø + ETA  * L2B 151Ø
C *    IS GREATER THAN 1.Ø IN DOUBLE PRECISION ARITHMETIC.            * L2B 152Ø
```

```
C *    FOR IBM COMPUTER TYPE, ETA = 16.**(-13)                          * L2B 153Ø
C *    FOR UNIVAC COMPUTER TYPE, ETA = 2.**(-59)                        * L2B 154Ø
C * 5. THE FOLLOWING FORTRAN FUNCTIONS SHOULD BE CHANGED --             * L2B 155Ø
C *      SINGLE PRECISION NAME     DOUBLE PRECISION NAME                * L2B 156Ø
C *            DBLE(X)                 X                               * L2B 157Ø
C *            FLOAT(N)                DBLE(FLOAT(N))                  * L2B 158Ø
C *            SQRT(X)                 DSQRT(X)                        * L2B 159Ø
C *    DBLE(X) IS USED IN SUBROUTINES DECOM2, SOLVE2, SOLVE3 AND        * L2B 16ØØ
C *       COVAR.                                                        * L2B 161Ø
C *    FLOAT(N) IS USED IN SUBROUTINE DECOM2.                           * L2B 162Ø
C *    SQRT(X) IS USED IN SUBROUTINE L2B.                               * L2B 163Ø
C * 6. IT MAY BE NECESSARY OR DESIRABLE TO CHANGE CERTAIN FORMATS IN    * L2B 164Ø
C *    SUBROUTINE ERROR, REPLACING G SPECIFICATIONS BY D.               * L2B 165Ø
C * 7. REPLACE STATEMENT L2B 25ØØ BY A STATEMENT READING                * L2B 166Ø
C *          K3 = 1                                                     * L2B 167Ø
C * 8. FURTHER DETAILS ARE GIVEN IN SUBROUTINE SOLVE2 IN CONNECTION     * L2B 168Ø
C *    WITH THE OMISSION OF ITERATIVE REFINEMENT.                       * L2B 169Ø
C * 9. IN SUBROUTINE L2B, STATEMENTS L2B 95Ø-1ØØØ, 182Ø-183Ø, 2Ø2Ø,     * L2B 17ØØ
C *    235Ø-236Ø, 248Ø-249Ø, 3Ø7Ø, 328Ø-357Ø AND 359Ø-362Ø SHOULD BE     * L2B 171Ø
C *    OMITTED.                                                         * L2B 172Ø
C *    STATEMENT NUMBERS GIVEN HERE REFER TO THOSE IN THE RIGHT-HAND    * L2B 173Ø
C *    MARGIN.  CERTAIN COMMENTS IN SUBROUTINE L2B DO NOT APPLY TO      * L2B 174Ø
C *    THE DOUBLE PRECISION VERSION.                                    * L2B 175Ø
C *                                                                     * L2B 176Ø
C ********************************************************************** L2B 177Ø
C                                                                        L2B 178Ø
      INTEGER IPIVOT(N)                                                  L2B 179Ø
      REAL A(MM,N), B(MM,L), C(1), ETA, QR(MMPNN,N),                     L2B 18ØØ
     * RES(MM,L), TOL, W(M), X(NN,L), Z                                 L2B 181Ø
      REAL DIGITX                                                        L2B 182Ø
      LOGICAL FAIL                                                       L2B 183Ø
      LOGICAL SING                                                       L2B 184Ø
C                                                                        L2B 185Ø
C SET VALUE OF ETA, A MACHINE-DEPENDENT PARAMETER.                       L2B 186Ø
C ETA, THE RELATIVE MACHINE PRECISION, IS THE SMALLEST POSITIVE REAL     L2B 187Ø
C NUMBER SUCH THAT 1.Ø + ETA IS GREATER THAN 1.Ø IN FLOATING-POINT       L2B 188Ø
C ARITHMETIC.                                                            L2B 189Ø
C                                                                        L2B 19ØØ
C FOR IBM COMPUTER TYPE, ETA = 16.**(-5)                                 L2B 191Ø
C FOR UNIVAC COMPUTER TYPE, ETA = 2.**(-26)                              L2B 192Ø
C FOR CDC COMPUTER TYPE, ETA = 2.**(-47)                                 L2B 193Ø
C FOR HONEYWELL COMPUTER TYPE, ETA = 2.**(-27)                           L2B 194Ø
C                                                                        L2B 195Ø
      ETA = 2.**(-26)                                                    L2B 196Ø
C                                                                        L2B 197Ø
C DEFAULT VALUE FOR TOL IS ZERO.                                         L2B 198Ø
C                                                                        L2B 199Ø
      IF (TOL.LT.Ø.Ø) TOL = Ø.Ø                                          L2B 2ØØØ
      IFAULT = Ø                                                         L2B 2Ø1Ø
      KSUM = Ø                                                           L2B 2Ø2Ø
C                                                                        L2B 2Ø3Ø
C PERFORM INITIAL CHECKING OF INPUT PARAMETERS, DIMENSIONS AND           L2B 2Ø4Ø
C WEIGHTS FOR POSSIBLE ERRORS.                                           L2B 2Ø5Ø
C                                                                        L2B 2Ø6Ø
      IF (M.GT.Ø .AND. N.GT.Ø .AND. L.GT.Ø) GO TO 1Ø                     L2B 2Ø7Ø
      IFAULT = 1                                                         L2B 2Ø8Ø
      CALL ERROR(IFAULT, K, Z)                                           L2B 2Ø9Ø
      RETURN                                                             L2B 21ØØ
   1Ø IF (M1.LE.M .AND. M1.LE.N .AND. M1.GE.Ø) GO TO 2Ø                  L2B 211Ø
      IFAULT = 2                                                         L2B 212Ø
      CALL ERROR(IFAULT, K, Z)                                           L2B 213Ø
      RETURN                                                             L2B 214Ø
   2Ø IF (M.LE.MM .AND. N.LE.NN .AND. M+N.LE.MMPNN) GO TO 3Ø             L2B 215Ø
      IFAULT = 3                                                         L2B 216Ø
      K = 2                                                              L2B 217Ø
      CALL ERROR(IFAULT, K, Z)                                           L2B 218Ø
      RETURN                                                             L2B 219Ø
   3Ø DO 4Ø I=1,M                                                        L2B 22ØØ
        IF (M1.GT.Ø .AND. I.LE.M1) W(I) = 1.Ø                            L2B 221Ø
        IF (W(I).GE.Ø.Ø) GO TO 4Ø                                        L2B 222Ø
        IFAULT = 4                                                       L2B 223Ø
        Z = W(I)                                                         L2B 224Ø
        CALL ERROR(IFAULT, I, Z)                                         L2B 225Ø
   4Ø CONTINUE                                                           L2B 226Ø
      IF (IFAULT.EQ.4) RETURN                                            L2B 227Ø
      DO 5Ø I=1,M                                                        L2B 228Ø
```

```
      W(I) = SQRT(W(I))                                                 L2B 229Ø
   5Ø CONTINUE                                                          L2B 23ØØ
C                                                                       L2B 231Ø
C SET PARAMETERS WHICH ALLOCATE VECTOR C TO CONTAIN CERTAIN FINAL       L2B 232Ø
C RESULTS AND ALSO TO BE USED AS WORK SPACE.                            L2B 233Ø
C                                                                       L2B 234Ø
C K1 IS STARTING POINT FOR NUMIT AND FAIL, OF LENGTH L.                 L2B 235Ø
C K2 IS STARTING POINT FOR DIGITX, OF LENGTH L.                         L2B 236Ø
C K3 IS STARTING POINT FOR D, OF LENGTH N.                              L2B 237Ø
C K4 IS STARTING POINT FOR K-TH COLUMN OF B, OF LENGTH M.               L2B 238Ø
C K5 IS STARTING POINT FOR K-TH COLUMN OF X, OF LENGTH N.               L2B 239Ø
C K6 IS STARTING POINT FOR K-TH COLUMN OF RES, OF LENGTH M.             L2B 24ØØ
C K7 IS STARTING POINT FOR WORK SPACE OF LENGTH M.                      L2B 241Ø
C K8 IS STARTING POINT FOR WORK SPACE OF LENGTH M.                      L2B 242Ø
C K9 IS STARTING POINT FOR WORK SPACE OF LENGTH N.                      L2B 243Ø
C K1Ø IS STARTING POINT FOR WORK SPACE OF LENGTH N.                     L2B 244Ø
C K11 IS STARTING POINT FOR WORK SPACE OF LENGTH M + N.                 L2B 245Ø
C K12 IS STARTING POINT FOR WORK SPACE OF LENGTH M + N.                 L2B 246Ø
C                                                                       L2B 247Ø
      K1 = 1                                                            L2B 248Ø
      K2 = K1 + L                                                       L2B 249Ø
      K3 = K2 + L                                                       L2B 25ØØ
      K4 = K3 + N                                                       L2B 251Ø
      K5 = K4 + M                                                       L2B 252Ø
      K6 = K5 + N                                                       L2B 253Ø
      K7 = K6 + M                                                       L2B 254Ø
      K8 = K7 + M                                                       L2B 255Ø
      K9 = K8 + M                                                       L2B 256Ø
      K1Ø = K9 + N                                                      L2B 257Ø
      K11 = K1Ø + N                                                     L2B 258Ø
      K12 = K11 + M + N                                                 L2B 259Ø
      K = K12 + M + N - 1                                               L2B 26ØØ
C                                                                       L2B 261Ø
C MULTIPLY EACH ROW OF MATRIX A (M BY N) BY ITS APPROPRIATE WEIGHT AND  L2B 262Ø
C STORE THE RESULT IN THE FIRST M ROWS OF ARRAY QR.  SET ARRAY C AND    L2B 263Ø
C THE LAST N ROWS OF ARRAY QR EQUAL TO ZERO.                            L2B 264Ø
C                                                                       L2B 265Ø
      DO 6Ø I=1,K                                                       L2B 266Ø
        C(I) = Ø.Ø                                                      L2B 267Ø
   6Ø CONTINUE                                                          L2B 268Ø
      MP1 = M + 1                                                       L2B 269Ø
      MPN = M + N                                                       L2B 27ØØ
      DO 9Ø J=1,N                                                       L2B 271Ø
        DO 7Ø I=1,M                                                     L2B 272Ø
          QR(I,J) = A(I,J)*W(I)                                         L2B 273Ø
   7Ø   CONTINUE                                                        L2B 274Ø
        DO 8Ø I=MP1,MPN                                                 L2B 275Ø
          QR(I,J) = Ø.Ø                                                 L2B 276Ø
   8Ø   CONTINUE                                                        L2B 277Ø
   9Ø CONTINUE                                                          L2B 278Ø
C                                                                       L2B 279Ø
C OBTAIN AN ORTHOGONAL DECOMPOSITION OF THE MATRIX STORED IN THE FIRST  L2B 28ØØ
C M ROWS OF ARRAY QR AND COMPUTE ITS RANK.                              L2B 281Ø
C                                                                       L2B 282Ø
      CALL DECOM2(M, N, M1, ETA, TOL, QR, C(K3), N1, IPIVOT, SING,      L2B 283Ø
     * MMPNN)                                                           L2B 284Ø
C                                                                       L2B 285Ø
      IF (.NOT.SING) GO TO 11Ø                                          L2B 286Ø
      IF (N1.GT.Ø) GO TO 1ØØ                                            L2B 287Ø
      IFAULT = 5                                                        L2B 288Ø
      CALL ERROR(IFAULT, K, Z)                                          L2B 289Ø
      RETURN                                                            L2B 29ØØ
  1ØØ IFAULT = 6                                                        L2B 291Ø
      CALL ERROR(IFAULT, K, Z)                                          L2B 292Ø
      RETURN                                                            L2B 293Ø
C                                                                       L2B 294Ø
C SEEK A SOLUTION (COEFFICIENTS AND RESIDUALS) FOR EACH OF THE L LEAST  L2B 295Ø
C SQUARES PROBLEMS WHOSE RIGHT-HAND SIDES ARE GIVEN IN THE ARRAY B.     L2B 296Ø
C                                                                       L2B 297Ø
  11Ø DO 2ØØ K=1,L                                                      L2B 298Ø
C K-TH RIGHT-HAND SIDE.                                                 L2B 299Ø
        KØ = K4 - 1                                                     L2B 3ØØØ
        DO 12Ø I=1,M                                                    L2B 3Ø1Ø
          KØ = KØ + 1                                                   L2B 3Ø2Ø
          C(KØ) = B(I,K)                                                L2B 3Ø3Ø
  12Ø   CONTINUE                                                        L2B 3Ø4Ø
```

```
C                                                                       L2B 3Ø5Ø
      CALL SOLVE2(M, N, M1, A, C(K4), W, N1, IPIVOT, QR, C(K3),         L2B 3Ø6Ø
     *  ETA, FAIL, NUMIT, DIGITX,                                       L2B 3Ø7Ø
     *  C(K5), C(K6), C(K7), C(K8), C(K9), C(K1Ø), C(K11), C(K12),      L2B 3Ø8Ø
     *  MM, MMPNN)                                                      L2B 3Ø9Ø
C                                                                       L2B 31ØØ
      KØ = K5 - 1                                                       L2B 311Ø
      DO 13Ø J=1,N                                                      L2B 312Ø
        KØ = KØ + 1                                                     L2B 313Ø
        X(J,K) = C(KØ)                                                  L2B 314Ø
 13Ø  CONTINUE                                                          L2B 315Ø
      IF (M1.EQ.Ø) GO TO 15Ø                                            L2B 316Ø
      DO 14Ø I=1,M1                                                     L2B 317Ø
        RES(I,K) = Ø.Ø                                                  L2B 318Ø
 14Ø  CONTINUE                                                          L2B 319Ø
 15Ø  M1P1 = M1 + 1                                                     L2B 32ØØ
      IF (M1P1.GT.M) GO TO 17Ø                                          L2B 321Ø
      KØ = K6 + M1 - 1                                                  L2B 322Ø
      DO 16Ø I=M1P1,M                                                   L2B 323Ø
        KØ = KØ + 1                                                     L2B 324Ø
        RES(I,K) = C(KØ)                                                L2B 325Ø
 16Ø  CONTINUE                                                          L2B 326Ø
 17Ø  CONTINUE                                                          L2B 327Ø
C                                                                       L2B 328Ø
C FOR RIGHT-HAND SIDES WHERE CONVERGENCE OF A SOLUTION IS REPORTED,     L2B 329Ø
C A CHECK IS MADE FOR EVIDENCE OF SEVERE ILL-CONDITIONING.  SUCH        L2B 33ØØ
C EVIDENCE IS FURNISHED BY LARGE VALUES OF NUMIT (NUMBER OF ITERATIONS  L2B 331Ø
C BEFORE CONVERGENCE WAS OBTAINED) AND SMALL VALUES OF DIGITX           L2B 332Ø
C (ESTIMATE OF THE NUMBER OF CORRECT DIGITS IN THE INITIAL SOLUTION     L2B 333Ø
C OF THE COEFFICIENTS).  IF NUMIT EXCEEDS -ALOG1Ø(ETA) A DIAGNOSTIC     L2B 334Ø
C MESSAGE IS PRINTED TO WARN OF ILL-CONDITIONING.  IF DIGITX IS LESS    L2B 335Ø
C THAN Ø.5 (HALF A DECIMAL DIGIT) A SIMILAR DIAGNOSTIC MESSAGE IS       L2B 336Ø
C PRINTED.                                                              L2B 337Ø
C                                                                       L2B 338Ø
      C(K) = FLOAT(NUMIT)                                               L2B 339Ø
      IF (FAIL) C(K) = -C(K)                                            L2B 34ØØ
      KØ = K2 + K - 1                                                   L2B 341Ø
      C(KØ) = DIGITX                                                    L2B 342Ø
      IF (.NOT.FAIL) GO TO 18Ø                                          L2B 343Ø
C KSUM IS A TALLY OF SOLUTIONS WHICH FAILED TO CONVERGE.                L2B 344Ø
      KSUM = KSUM + 1                                                   L2B 345Ø
      IFAULT = 8                                                        L2B 346Ø
      CALL ERROR(IFAULT, K, Z)                                          L2B 347Ø
      GO TO 2ØØ                                                         L2B 348Ø
 18Ø  Z = -ALOG1Ø(ETA)                                                  L2B 349Ø
      IF (FLOAT(NUMIT).LE.Z) GO TO 19Ø                                  L2B 35ØØ
      IFAULT = 9                                                        L2B 351Ø
      Z = FLOAT(NUMIT)                                                  L2B 352Ø
      CALL ERROR(IFAULT, K, Z)                                          L2B 353Ø
 19Ø  IF (DIGITX.GE.Ø.5) GO TO 2ØØ                                      L2B 354Ø
      IFAULT = 1Ø                                                       L2B 355Ø
      Z = DIGITX                                                        L2B 356Ø
      CALL ERROR(IFAULT, K, Z)                                          L2B 357Ø
 2ØØ CONTINUE                                                           L2B 358Ø
     IF (KSUM.LT.L) GO TO 21Ø                                           L2B 359Ø
     IFAULT = 7                                                         L2B 36ØØ
     CALL ERROR(IFAULT, K, Z)                                           L2B 361Ø
     RETURN                                                             L2B 362Ø
 21Ø IF (N1.LT.N) RETURN                                                L2B 363Ø
     DO 23Ø I=1,N                                                       L2B 364Ø
       MPI = M + I                                                      L2B 365Ø
       DO 22Ø J=1,N                                                     L2B 366Ø
         QR(I,J) = QR(MPI,J)                                            L2B 367Ø
 22Ø   CONTINUE                                                         L2B 368Ø
       QR(I,I) = Ø.Ø                                                    L2B 369Ø
 23Ø CONTINUE                                                           L2B 37ØØ
C                                                                       L2B 371Ø
C COMPUTE THE UNSCALED COVARIANCE MATRIX OF THE COEFFICIENTS.           L2B 372Ø
C                                                                       L2B 373Ø
     CALL COVAR(N, M1, N1, IPIVOT, QR, C(K3), C(K9), MMPNN)             L2B 374Ø
C                                                                       L2B 375Ø
C IN CERTAIN PROBLEMS, SOME DIAGONAL TERMS OF THE UNSCALED COVARIANCE   L2B 376Ø
C MATRIX ARE EQUAL TO ZERO OR TO SMALL POSITIVE NUMBERS.  BECAUSE OF    L2B 377Ø
C ROUNDING ERRORS, COMPUTED VALUES FOR THESE TERMS MAY BE SMALL         L2B 378Ø
C NEGATIVE NUMBERS.  AN ERROR DIAGNOSTIC IS PRINTED IF ANY DIAGONAL     L2B 379Ø
C TERM IS NEGATIVE.                                                     L2B 38ØØ
```

```
C                                                                       L2B 381Ø
      DO 24Ø J=1,N                                                      L2B 382Ø
        IF (QR(J,J).GE.Ø.Ø) GO TO 24Ø                                   L2B 383Ø
        IFAULT = 11                                                     L2B 384Ø
        Z = QR(J,J)                                                     L2B 385Ø
        CALL ERROR(IFAULT, J, Z)                                        L2B 386Ø
  24Ø CONTINUE                                                          L2B 387Ø
      RETURN                                                            L2B 388Ø
      END                                                               L2B 389Ø


C     SUBROUTINE DECOM1(...)                                            DC1   1Ø
C SUBROUTINE DECOM1 USES A MODIFIED GRAM-SCHMIDT ALGORITHM WITH         DC1   2Ø
C PIVOTING TO OBTAIN AN ORTHOGONAL DECOMPOSITION OF THE INPUT MATRIX    DC1   3Ø
C GIVEN IN Q.                                                           DC1   4Ø
C THE INPUT PARAMETER TOL (EQUAL EITHER TO ZERO OR TO A POSITIVE        DC1   5Ø
C NUMBER) IS USED IN DETERMINING THE RANK OF MATRIX Q.                  DC1   6Ø
C NOTE --                                                               DC1   7Ø
C     (1) IF TOL EQUALS ZERO, THE TOLERANCE USED AT STATEMENT DC1 1Ø8Ø   DC1   8Ø
C         WILL BE BASED ON MACHINE PRECISION.                           DC1   9Ø
C         UNDER THIS APPROACH, THE TOLERANCE (TOL2) IS SET EQUAL TO     DC1  1ØØ
C         (FLOAT(N)*ETA)**2*D(M1+1) AT STATEMENT DC1 1Ø7Ø.               DC1  11Ø
C         IF DESIRED, THE USER CAN OBTAIN A MORE CONSERVATIVE           DC1  12Ø
C         TOLERANCE BY REPLACING N IN THIS STATEMENT BY A LARGER        DC1  13Ø
C         QUANTITY.                                                     DC1  14Ø
C     (2) IF TOL IS GREATER THAN ZERO, TOL2 (EQUAL TO THE SQUARE OF     DC1  15Ø
C         TOL) WILL BE USED AT STATEMENT DC1 1Ø8Ø AS AN ABSOLUTE        DC1  16Ø
C         TOLERANCE FOR COMPARISON WITH DIAGONAL ELEMENTS OF THE        DC1  17Ø
C         TRIANGULAR MATRIX OBTAINED IN THE DECOMPOSITION.  UNDER THIS  DC1  18Ø
C         APPROACH, THE VALUE OF TOL CAN BE BASED ON KNOWLEDGE          DC1  19Ø
C         CONCERNING THE ACCURACY OF THE DATA.                          DC1  2ØØ
C ON EXIT, THE ARRAYS Q, R, D AND IPIVOT CONTAIN THE RESULTS OF THE     DC1  21Ø
C DECOMPOSITION WHICH ARE NEEDED FOR OBTAINING AN INITIAL SOLUTION      DC1  22Ø
C AND FOR ITERATIVE REFINEMENT.                                         DC1  23Ø
C ON EXIT, N1 IS THE COMPUTED RANK OF THE INPUT MATRIX Q.               DC1  24¢
C ON EXIT, SING IS SET EQUAL TO .TRUE. WHENEVER                         DC1  25Ø
C     (1) N1 = Ø (I.E., INPUT MATRIX Q EQUALS ZERO OR MATRIX OF         DC1  26Ø
C         CONSTRAINTS EQUALS ZERO), OR                                  DC1  27Ø
C     (2) N1 IS LESS THAN M1 (I.E., THE M1 BY N MATRIX OF LINEAR        DC1  28Ø
C         CONSTRAINTS IS SINGULAR).                                     DC1  29Ø
C OTHERWISE, ON EXIT FROM DECOM1, SING = .FALSE.                        DC1  3ØØ
C ON EXIT, THE VECTOR IPIVOT RECORDS THE ORDER IN WHICH THE COLUMNS     DC1  31Ø
C OF Q WERE SELECTED BY THE PIVOTING SCHEME IN THE COURSE OF THE        DC1  32Ø
C ORTHOGONAL DECOMPOSITION.                                             DC1  33Ø
      SUBROUTINE DECOM1(M, N, M1, ETA, TOL, Q, R, D, N1, IPIVOT, SING,  DC1  34Ø
     * MM, NN)                                                          DC1  35Ø
      INTEGER IPIVOT(N)                                                 DC1  36Ø
      REAL C, D(1), DM, DS, ETA, Q(MM,N), R(NN,N), RSJ, TOL, TOL2       DC1  37Ø
      DOUBLE PRECISION SUM                                              DC1  38Ø
      LOGICAL FSUM, SING                                                DC1  39Ø
      N1 = Ø                                                            DC1  4Ø¢
      SING = .TRUE.                                                     DC1  41Ø
      FSUM = .TRUE.                                                     DC1  42Ø
      MV = 1                                                            DC1  43Ø
      MH = M1                                                           DC1  44Ø
      IF (TOL.GT.Ø.Ø) TOL2 = TOL**2                                     DC1  45Ø
      DO 1Ø J=1,N                                                       DC1  46Ø
        D(J) = Ø.Ø                                                      DC1  47Ø
        IPIVOT(J) = J                                                   DC1  48Ø
   1Ø CONTINUE                                                          DC1  49Ø
C STEP NUMBER NS OF THE DECOMPOSITION.                                  DC1  5ØØ
      DO 21Ø NS=1,N                                                     DC1  51Ø
        NSM1 = NS - 1                                                   DC1  52Ø
        NSP1 = NS + 1                                                   DC1  53Ø
        IF (NS.EQ.M1+1) GO TO 2Ø                                        DC1  54Ø
        GO TO 3Ø                                                        DC1  55Ø
   2Ø   IF (M1.EQ.M) GO TO 15Ø                                          DC1  56Ø
        MV = M1 + 1                                                     DC1  57Ø
        MH = M                                                          DC1  58Ø
        FSUM = .TRUE.                                                   DC1  59Ø
C PIVOT SEARCH.                                                         DC1  6ØØ
   3Ø   DS = Ø.Ø                                                        DC1  61Ø
        NP = NS                                                         DC1  62Ø
        DO 8Ø J=NS,N                                                    DC1  63Ø
          IK = IPIVOT(J)                                                DC1  64Ø
          IF (FSUM) GO TO 4Ø                                            DC1  65Ø
```

```
            GO TO 6Ø                                                    DC1  66Ø
     4Ø     SUM = Ø.Ø                                                   DC1  67Ø
            DO 5Ø L=MV,MH                                               DC1  68Ø
              SUM = SUM + DBLE(Q(L,IK))*DBLE(Q(L,IK))                   DC1  69Ø
     5Ø     CONTINUE                                                    DC1  7ØØ
            D(J) = SUM                                                  DC1  71Ø
     6Ø     IF (DS.LT.D(J)) GO TO 7Ø                                    DC1  72Ø
            GO TO 8Ø                                                    DC1  73Ø
     7Ø     DS = D(J)                                                   DC1  74Ø
            NP = J                                                      DC1  75Ø
     8Ø   CONTINUE                                                      DC1  76Ø
C END PIVOT SEARCH.                                                     DC1  77Ø
          IK = IPIVOT(NP)                                               DC1  78Ø
          IF (FSUM) DM = DS                                             DC1  79Ø
          IF (DS.LT.ETA*DM) GO TO 9Ø                                    DC1  8ØØ
          FSUM = .FALSE.                                                DC1  81Ø
          GO TO 1ØØ                                                     DC1  82Ø
     9Ø   FSUM = .TRUE.                                                 DC1  83Ø
    1ØØ   IF (FSUM) GO TO 3Ø                                            DC1  84Ø
          IF (NP.NE.NS) GO TO 11Ø                                       DC1  85Ø
          GO TO 13Ø                                                     DC1  86Ø
C COLUMN INTERCHANGE.                                                   DC1  87Ø
    11Ø   IPIVOT(NP) = IPIVOT(NS)                                       DC1  88Ø
          IPIVOT(NS) = IK                                               DC1  89Ø
          D(NP) = D(NS)                                                 DC1  9ØØ
          IF (NS.EQ.1) GO TO 13Ø                                        DC1  91Ø
          DO 12Ø L=1,NSM1                                               DC1  92Ø
            C = R(L,NP)                                                 DC1  93Ø
            R(L,NP) = R(L,NS)                                           DC1  94Ø
            R(L,NS) = C                                                 DC1  95Ø
    12Ø   CONTINUE                                                      DC1  96Ø
C END COLUMN INTERCHANGE.                                               DC1  97Ø
C RETURN HERE IF N1 = Ø.  EITHER INPUT MATRIX Q EQUALS ZERO OR MATRIX   DC1  98Ø
C OF CONSTRAINTS EQUALS ZERO.                                           DC1  99Ø
    13Ø   IF (NS.EQ.1 .AND. DS.EQ.Ø.Ø) RETURN                           DC1 1ØØØ
          SUM = Ø.Ø                                                     DC1 1Ø1Ø
          DO 14Ø L=MV,MH                                                DC1 1Ø2Ø
            SUM = SUM + DBLE(Q(L,IK))*DBLE(Q(L,IK))                     DC1 1Ø3Ø
    14Ø   CONTINUE                                                      DC1 1Ø4Ø
          D(NS) = SUM                                                   DC1 1Ø5Ø
          DS = D(NS)                                                    DC1 1Ø6Ø
          IF (TOL.EQ.Ø.Ø) TOL2 = (FLOAT(N)*ETA)**2*D(M1+1)              DC1 1Ø7Ø
          IF (NS.GT.M1 .AND. DS.LE.TOL2) GO TO 15Ø                      DC1 1Ø8Ø
          GO TO 16Ø                                                     DC1 1Ø9Ø
    15Ø   SING = .FALSE.                                                DC1 11ØØ
C RETURN HERE IF N1.LT.N, N1.GT.Ø AND N1.GE.M1.                         DC1 111Ø
          RETURN                                                        DC1 112Ø
C RETURN HERE IF MATRIX OF CONSTRAINTS IS FOUND TO BE SINGULAR.         DC1 113Ø
    16Ø   IF (DS.EQ.Ø.Ø) RETURN                                         DC1 114Ø
          IF (NSP1.GT.N) GO TO 2ØØ                                      DC1 115Ø
C BEGIN ORTHOGONALIZATION.                                              DC1 116Ø
          DO 19Ø J=NSP1,N                                               DC1 117Ø
            NP = IPIVOT(J)                                              DC1 118Ø
            SUM = Ø.Ø                                                   DC1 119Ø
            DO 17Ø L=MV,MH                                              DC1 12ØØ
              SUM = SUM + DBLE(Q(L,NP))*DBLE(Q(L,IK))                   DC1 121Ø
    17Ø     CONTINUE                                                    DC1 122Ø
            RSJ = SUM                                                   DC1 123Ø
            RSJ = RSJ/DS                                                DC1 124Ø
            R(NS,J) = RSJ                                               DC1 125Ø
            DO 18Ø L=MV,M                                               DC1 126Ø
              Q(L,NP) = Q(L,NP) - RSJ*Q(L,IK)                           DC1 127Ø
    18Ø     CONTINUE                                                    DC1 128Ø
            D(J) = D(J) - DS*RSJ*RSJ                                    DC1 129Ø
    19Ø   CONTINUE                                                      DC1 13ØØ
C END ORTHOGONALIZATION.                                                DC1 131Ø
    2ØØ   N1 = N1 + 1                                                   DC1 132Ø
    21Ø CONTINUE                                                        DC1 133Ø
C END STEP NUMBER NS.                                                   DC1 134Ø
        SING = .FALSE.                                                  DC1 135Ø
C NORMAL RETURN.  N1 = N.                                               DC1 136Ø
        RETURN                                                          DC1 137Ø
        END                                                             DC1 138Ø
```

```
C     SUBROUTINE DECOM2(...)                                              DC2   1Ø
C SUBROUTINE DECOM2 USES A MODIFIED GRAM-SCHMIDT ALGORITHM WITH           DC2   2Ø
C PIVOTING TO OBTAIN AN ORTHOGONAL DECOMPOSITION OF THE INPUT MATRIX      DC2   3Ø
C GIVEN IN QR.                                                            DC2   4Ø
C THE INPUT PARAMETER TOL (EQUAL EITHER TO ZERO OR TO A POSITIVE          DC2   5Ø
C NUMBER) IS USED IN DETERMINING THE RANK OF MATRIX QR.                   DC2   6Ø
C NOTE --                                                                 DC2   7Ø
C     (1) IF TOL EQUALS ZERO, THE TOLERANCE USED AT STATEMENT DC2 118Ø    DC2   8Ø
C         WILL BE BASED ON MACHINE PRECISION.                             DC2   9Ø
C         UNDER THIS APPROACH, THE TOLERANCE (TOL2) IS SET EQUAL TO       DC2  1ØØ
C         (FLOAT(N)*ETA)**2*D(M1+1) AT STATEMENT DC2 117Ø.                DC2  11Ø
C         IF DESIRED, THE USER CAN OBTAIN A MORE CONSERVATIVE             DC2  12Ø
C         TOLERANCE BY REPLACING N IN THIS STATEMENT BY A LARGER          DC2  13Ø
C         QUANTITY.                                                       DC2  14Ø
C     (2) IF TOL IS GREATER THAN ZERO, TOL2 (EQUAL TO THE SQUARE OF       DC2  15Ø
C         TOL) WILL BE USED AT STATEMENT DC2 118Ø AS AN ABSOLUTE          DC2  16Ø
C         TOLERANCE FOR COMPARISON WITH DIAGONAL ELEMENTS OF THE          DC2  17Ø
C         TRIANGULAR MATRIX OBTAINED IN THE DECOMPOSITION.  UNDER THIS    DC2  18Ø
C         APPROACH, THE VALUE OF TOL CAN BE BASED ON KNOWLEDGE            DC2  19Ø
C         CONCERNING THE ACCURACY OF THE DATA.                            DC2  2ØØ
C ON EXIT, THE ARRAYS QR, D AND IPIVOT CONTAIN THE RESULTS OF THE         DC2  21Ø
C DECOMPOSITION WHICH ARE NEEDED FOR OBTAINING AN INITIAL SOLUTION        DC2  22Ø
C AND FOR ITERATIVE REFINEMENT.                                           DC2  23Ø
C ON EXIT, N1 IS THE COMPUTED RANK OF THE INPUT MATRIX QR.                DC2  24Ø
C ON EXIT, SING IS SET EQUAL TO .TRUE. WHENEVER                           DC2  25Ø
C     (1) N1 = Ø (I.E., INPUT MATRIX QR EQUALS ZERO OR MATRIX OF          DC2  26Ø
C         CONSTRAINTS EQUALS ZERO), OR                                    DC2  27Ø
C     (2) N1 IS LESS THAN M1 (I.E., THE M1 BY N MATRIX OF LINEAR          DC2  28Ø
C         CONSTRAINTS IS SINGULAR).                                       DC2  29Ø
C OTHERWISE, ON EXIT FROM DECOM2, SING = .FALSE.                          DC2  3ØØ
C ON EXIT, THE VECTOR IPIVOT RECORDS THE ORDER IN WHICH THE COLUMNS       DC2  31Ø
C OF QR WERE SELECTED BY THE PIVOTING SCHEME IN THE COURSE OF THE         DC2  32Ø
C ORTHOGONAL DECOMPOSITION.                                               DC2  33Ø
      SUBROUTINE DECOM2(M, N, M1, ETA, TOL, QR, D, N1, IPIVOT, SING,      DC2  34Ø
     * MMPNN)                                                            DC2  35Ø
      INTEGER IPIVOT(N)                                                   DC2  36Ø
      REAL C, D(1), DM, DS, ETA, QR(MMPNN,N), RSJ, TOL, TOL2              DC2  37Ø
      DOUBLE PRECISION SUM                                                DC2  38Ø
      LOGICAL FINIS, FSUM, SING                                           DC2  39Ø
      N1 = Ø                                                              DC2  4ØØ
      SING = .TRUE.                                                       DC2  41Ø
      FSUM = .TRUE.                                                       DC2  42Ø
      MV = 1                                                              DC2  43Ø
      MH = M1                                                             DC2  44Ø
      MS = M                                                              DC2  45Ø
      MP1 = M + 1                                                         DC2  46Ø
      FINIS = .FALSE.                                                     DC2  47Ø
      IF (TOL.GT.Ø.Ø) TOL2 = TOL**2                                       DC2  48Ø
      DO 1Ø J=1,N                                                         DC2  49Ø
        D(J) = Ø.Ø                                                        DC2  5ØØ
        IPIVOT(J) = J                                                     DC2  51Ø
   1Ø CONTINUE                                                            DC2  52Ø
C STEP NUMBER NS OF THE DECOMPOSITION.                                    DC2  53Ø
      DO 35Ø NS=1,N                                                       DC2  54Ø
        K = M + NS                                                        DC2  55Ø
        IF (NS.EQ.M1+1) GO TO 2Ø                                          DC2  56Ø
        GO TO 3Ø                                                          DC2  57Ø
   2Ø   IF (M1.EQ.M) GO TO 2ØØ                                            DC2  58Ø
        MV = M1 + 1                                                       DC2  59Ø
        MH = M                                                            DC2  6ØØ
        FSUM = .TRUE.                                                     DC2  61Ø
   3Ø   IF (.NOT.FINIS) GO TO 4Ø                                          DC2  62Ø
        GO TO 15Ø                                                         DC2  63Ø
C PIVOT SEARCH.                                                           DC2  64Ø
   4Ø   DS = Ø.Ø                                                          DC2  65Ø
        NP = NS                                                           DC2  66Ø
        DO 9Ø J=NS,N                                                      DC2  67Ø
          IF (FSUM) GO TO 5Ø                                              DC2  68Ø
          GO TO 7Ø                                                        DC2  69Ø
   5Ø     SUM = Ø.Ø                                                      DC2  7ØØ
          DO 6Ø L=MV,MH                                                   DC2  71Ø
            SUM = SUM + DBLE(QR(L,J))*DBLE(QR(L,J))                       DC2  72Ø
   6Ø     CONTINUE                                                       DC2  73Ø
          D(J) = SUM                                                      DC2  74Ø
   7Ø     IF (DS.LT.D(J)) GO TO 8Ø                                       DC2  75Ø
          GO TO 9Ø                                                        DC2  76Ø
```

```
   8Ø      DS = D(J)                                                    DC2  77Ø
           NP = J                                                       DC2  78Ø
   9Ø    CONTINUE                                                       DC2  79Ø
         IF (FSUM) DM = DS                                            DC2  8ØØ
         IF (DS.LT.ETA*DM) GO TO 1ØØ                                  DC2  81Ø
         FSUM = .FALSE.                                               DC2  82Ø
         GO TO 11Ø                                                    DC2  83Ø
  1ØØ    FSUM = .TRUE.                                                DC2  84Ø
  11Ø    IF (FSUM) GO TO 4Ø                                           DC2  85Ø
         IF (NP.NE.NS) GO TO 12Ø                                      DC2  86Ø
         GO TO 14Ø                                                    DC2  87Ø
C COLUMN INTERCHANGE.                                                 DC2  88Ø
  12Ø    IK = IPIVOT(NP)                                              DC2  89Ø
         IPIVOT(NP) = IPIVOT(NS)                                      DC2  9ØØ
         IPIVOT(NS) = IK                                              DC2  91Ø
         D(NP) = D(NS)                                                DC2  92Ø
         KM1 = K - 1                                                  DC2  93Ø
         DO 13Ø L=1,KM1                                               DC2  94Ø
           C = QR(L,NP)                                               DC2  95Ø
           QR(L,NP) = QR(L,NS)                                        DC2  96Ø
           QR(L,NS) = C                                               DC2  97Ø
  13Ø    CONTINUE                                                     DC2  98Ø
C END COLUMN INTERCHANGE.                                             DC2  99Ø
C END PIVOT SEARCH.                                                   DC2 1ØØØ
C RETURN HERE IF N1 = Ø.  EITHER INPUT MATRIX QR EQUALS ZERO OR       DC2 1Ø1Ø
C MATRIX OF CONSTRAINTS EQUALS ZERO.                                  DC2 1Ø2Ø
  14Ø    IF (NS.EQ.1 .AND. DS.EQ.Ø.Ø) RETURN                          DC2 1Ø3Ø
         GO TO 16Ø                                                    DC2 1Ø4Ø
  15Ø    MS = K - 1                                                   DC2 1Ø5Ø
         MH = K - 1                                                   DC2 1Ø6Ø
  16Ø    IF (FINIS) GO TO 17Ø                                         DC2 1Ø7Ø
         C = Ø.Ø                                                      DC2 1Ø8Ø
         GO TO 18Ø                                                    DC2 1Ø9Ø
  17Ø    C = 1.Ø                                                      DC2 11ØØ
  18Ø    SUM = DBLE(C)                                                DC2 111Ø
         DO 19Ø L=MV,MH                                               DC2 112Ø
           SUM = SUM + DBLE(QR(L,NS))*DBLE(QR(L,NS))                  DC2 113Ø
  19Ø    CONTINUE                                                     DC2 114Ø
         D(NS) = SUM                                                  DC2 115Ø
         DS = D(NS)                                                   DC2 116Ø
         IF (TOL.EQ.Ø.Ø) TOL2 = (FLOAT(N)*ETA)**2*D(M1+1)             DC2 117Ø
         IF (.NOT.FINIS .AND. NS.GT.M1 .AND. DS.LE.TOL2) GO TO 2ØØ    DC2 118Ø
         GO TO 29Ø                                                    DC2 119Ø
  2ØØ    FINIS = .TRUE.                                               DC2 12ØØ
         MV = M + 1                                                   DC2 121Ø
         DO 28Ø NP=NS,N                                               DC2 122Ø
           IF (1.GT.M1) GO TO 25Ø                                     DC2 123Ø
           DO 21Ø L=1,M1                                              DC2 124Ø
             QR(L,NP) = Ø.Ø                                           DC2 125Ø
  21Ø      CONTINUE                                                   DC2 126Ø
           DO 24Ø J=1,M1                                              DC2 127Ø
             SUM = Ø.Ø                                                DC2 128Ø
             DO 22Ø L=1,M                                             DC2 129Ø
               SUM = SUM + DBLE(QR(L,J))*DBLE(QR(L,NP))               DC2 13ØØ
  22Ø        CONTINUE                                                 DC2 131Ø
             C = SUM                                                  DC2 132Ø
             C = C/D(J)                                               DC2 133Ø
             DO 23Ø L=1,M1                                            DC2 134Ø
               QR(L,NP) = QR(L,NP) - C*QR(L,J)                        DC2 135Ø
  23Ø        CONTINUE                                                 DC2 136Ø
  24Ø      CONTINUE                                                   DC2 137Ø
  25Ø      MPN1 = M + N1                                              DC2 138Ø
           DO 27Ø JJ=MP1,MPN1                                         DC2 139Ø
             J = (M + 1) + (M + N1) - JJ                              DC2 14ØØ
             SUM = Ø.Ø                                                DC2 141Ø
             DO 26Ø L=J,MPN1                                          DC2 142Ø
               LMM = L - M                                            DC2 143Ø
               SUM = SUM + DBLE(QR(J,LMM))*DBLE(QR(L,NP))             DC2 144Ø
  26Ø        CONTINUE                                                 DC2 145Ø
             QR(J,NP) = -SUM                                          DC2 146Ø
  27Ø      CONTINUE                                                   DC2 147Ø
  28Ø    CONTINUE                                                     DC2 148Ø
         GO TO 15Ø                                                    DC2 149Ø
C RETURN HERE IF MATRIX OF CONSTRAINTS IS FOUND TO BE SINGULAR.       DC2 15ØØ
  29Ø    IF (DS.EQ.Ø.Ø) RETURN                                        DC2 151Ø
         QR(K,NS) = -1.Ø                                              DC2 152Ø
```

```
      NSP1 = NS + 1                                                    DC2 153Ø
      IF (NSP1.GT.N) GO TO 34Ø                                         DC2 154Ø
C BEGIN ORTHOGONALIZATION.                                             DC2 155Ø
      DO 33Ø J=NSP1,N                                                  DC2 156Ø
        SUM = Ø.Ø                                                      DC2 157Ø
        DO 3ØØ L=MV,MH                                                 DC2 158Ø
          SUM = SUM + DBLE(QR(L,J))*DBLE(QR(L,NS))                     DC2 159Ø
 3ØØ    CONTINUE                                                       DC2 16ØØ
        RSJ = SUM                                                      DC2 161Ø
        RSJ = RSJ/DS                                                   DC2 162Ø
        QR(K,J) = RSJ                                                  DC2 163Ø
        DO 31Ø L=1,MS                                                  DC2 164Ø
          QR(L,J) = QR(L,J) - RSJ*QR(L,NS)                             DC2 165Ø
 31Ø    CONTINUE                                                       DC2 166Ø
        IF (.NOT.FINIS) GO TO 32Ø                                      DC2 167Ø
        GO TO 33Ø                                                      DC2 168Ø
 32Ø    D(J) = D(J) - DS*RSJ*RSJ                                       DC2 169Ø
 33Ø  CONTINUE                                                         DC2 17ØØ
C END ORTHOGONALIZATION.                                               DC2 171Ø
 34Ø   IF (.NOT.FINIS) N1 = N1 + 1                                     DC2 172Ø
 35Ø CONTINUE                                                          DC2 173Ø
C END STEP NUMBER NS.                                                  DC2 174Ø
      SING = .FALSE.                                                   DC2 175Ø
C NORMAL RETURN.                                                       DC2 176Ø
      RETURN                                                           DC2 177Ø
      END                                                              DC2 178Ø


C     SUBROUTINE SOLVE1(...)                                           SV1   1Ø
C SUBROUTINE SOLVE1 USES THE ORTHOGONAL DECOMPOSITION STORED IN Q, R,   SV1   2Ø
C D AND IPIVOT TO COMPUTE THE SOLUTION (COEFFICIENTS AND RESIDUALS)     SV1   3Ø
C TO THE LEAST SQUARES PROBLEM WHOSE RIGHT-HAND SIDE IS GIVEN IN B.     SV1   4Ø
C IN THE EVENT THAT N1 (THE COMPUTED RANK OF MATRIX H) IS LESS THAN N   SV1   5Ø
C (THE NUMBER OF UNKNOWN COEFFICIENTS), THE ORIGINAL MATRIX H (M BY N)  SV1   6Ø
C IS REPLACED BY A SMALLER MATRIX (M BY N1) WHOSE COLUMNS ARE LINEARLY  SV1   7Ø
C INDEPENDENT, AND A SOLUTION IS SOUGHT FOR THE SMALLER SYSTEM OF       SV1   8Ø
C EQUATIONS.  THUS N - N1 COLUMNS OF THE ORIGINAL MATRIX H ARE DELETED, SV1   9Ø
C AND COEFFICIENTS CORRESPONDING TO THESE N - N1 DELETED COLUMNS WILL   SV1  1ØØ
C BE SET EQUAL TO ZERO.                                                SV1  11Ø
C IN NORMAL EXITS, THE SOLUTION IS CONTAINED IN THE VECTOR X           SV1  12Ø
C (COEFFICIENTS) AND THE VECTOR RES (RESIDUALS).                       SV1  13Ø
C ITERATIVE REFINEMENT IS USED TO IMPROVE THE ACCURACY OF THE INITIAL   SV1  14Ø
C SOLUTION.                                                            SV1  15Ø
C ON EXIT, FAIL IS SET EQUAL TO .TRUE. IF THE SOLUTION FAILS TO        SV1  16Ø
C IMPROVE SUFFICIENTLY.  OTHERWISE, FAIL = .FALSE.  INFORMATION ON THE SV1  17Ø
C BEHAVIOR OF THE ITERATIVE REFINEMENT PROCEDURE IS GIVEN BY NUMIT AND SV1  18Ø
C DIGITX.  NUMIT IS THE NUMBER OF ITERATIONS CARRIED OUT IN ATTEMPTING SV1  19Ø
C TO OBTAIN A SOLUTION.  DIGITX IS AN ESTIMATE OF THE NUMBER OF        SV1  2ØØ
C CORRECT DIGITS IN THE INITIAL SOLUTION OF THE COEFFICIENTS.          SV1  21Ø
C                                                                      SV1  22Ø
C ********* CONVERSION OF THIS SUBROUTINE TO DOUBLE PRECISION ********* SV1  23Ø
C * IF THE PROGRAM IS CONVERTED SO THAT ALL CALCULATIONS ARE DONE IN  * SV1  24Ø
C * DOUBLE PRECISION ARITHMETIC, THE ITERATIVE REFINEMENT PRESENTLY   * SV1  25Ø
C * INCLUDED IN SOLVE1 SHOULD BE OMITTED, SINCE THE SUCCESS OF THIS   * SV1  26Ø
C * PROCEDURE DEPENDS ON COMPUTING INNER PRODUCTS IN GREATER          * SV1  27Ø
C * PRECISION THAN OTHER CALCULATIONS.                               * SV1  28Ø
C * SEE COMMENTS IN SUBROUTINE L2A REGARDING CONVERSION TO DOUBLE     * SV1  29Ø
C * PRECISION.  IN ADDITION, THE FOLLOWING COMMENTS INDICATE HOW TO   * SV1  3ØØ
C * OMIT THE ITERATIVE REFINEMENT FROM THIS SUBROUTINE.  STATEMENT    * SV1  31Ø
C * NUMBERS GIVEN HERE REFER TO THOSE IN THE RIGHT-HAND MARGIN.       * SV1  32Ø
C *                                                                   * SV1  33Ø
C * 1. IN STATEMENT SV1 48Ø CHANGE REAL TO DOUBLE PRECISION.          * SV1  34Ø
C * 2. REPLACE STATEMENT SV1 81Ø BY A STATEMENT READING               * SV1  35Ø
C *       3Ø DO 5Ø I=1,M                                              * SV1  36Ø
C * 3. AFTER STATEMENT SV1 1Ø5Ø INSERT A STATEMENT READING            * SV1  37Ø
C *          RETURN                                                   * SV1  38Ø
C * 4. OMIT STATEMENTS SV1 14Ø-21Ø, 45Ø, 5ØØ-51Ø, 53Ø-56Ø, 61Ø,       * SV1  39Ø
C *    68Ø-78Ø, 16ØØ-178Ø AND 18ØØ-186Ø.                              * SV1  4ØØ
C *                                                                   * SV1  41Ø
C ********************************************************************* SV1  42Ø
C                                                                      SV1  43Ø
      SUBROUTINE SOLVE1(M, N, M1, A, B, W, N1, IPIVOT, Q, R, D,        SV1  44Ø
     * ETA, FAIL, NUMIT, DIGITX,                                       SV1  45Ø
     * X, RES, F, WRES, G, Y, MM, NN)                                  SV1  46Ø
      INTEGER IPIVOT(N)                                                SV1  47Ø
      REAL A(MM,N), B(1), C, D(1), F(1), G(1), Q(MM,N),                SV1  48Ø
```

```
     * R(NN,N), RES(1), W(M), WRES(1), X(1), Y(1)                    SV1  49Ø
      REAL DIGITX, DXNORM, ETA, ETA2, RDR1, RDR2, RDX1, RDX2, RNR,  SV1  5ØØ
     * RNX, XNORM                                                    SV1  51Ø
      DOUBLE PRECISION SUM                                          SV1  52Ø
      LOGICAL FAIL                                                  SV1  53Ø
      NUMIT = Ø                                                     SV1  54Ø
      KZ = Ø                                                        SV1  55Ø
      ETA2 = ETA*ETA                                                SV1  56Ø
      DO 1Ø I=1,M                                                   SV1  57Ø
        F(I) = B(I)*W(I)                                            SV1  58Ø
        WRES(I) = Ø.Ø                                               SV1  59Ø
        RES(I) = Ø.Ø                                                SV1  6ØØ
        IF (W(I).EQ.Ø.Ø) KZ = KZ + 1                                SV1  61Ø
   1Ø CONTINUE                                                      SV1  62Ø
      DO 2Ø J=1,N                                                   SV1  63Ø
        X(J) = Ø.Ø                                                  SV1  64Ø
        G(J) = Ø.Ø                                                  SV1  65Ø
   2Ø CONTINUE                                                      SV1  66Ø
      K = Ø                                                         SV1  67Ø
      RDX2 = Ø.Ø                                                    SV1  68Ø
      RDR2 = Ø.Ø                                                    SV1  69Ø
C BEGIN K-TH ITERATION STEP.                                        SV1  7ØØ
   3Ø IF (K.LT.2) GO TO 4Ø                                          SV1  71Ø
      IF (((64.*RDX2.LT.RDX1) .AND. (RDX2.GT.ETA2*RNX)) .OR.        SV1  72Ø
     * ((64.*RDR2.LT.RDR1) .AND. (RDR2.GT.ETA2*RNR))) GO TO 4Ø     SV1  73Ø
      GO TO 3ØØ                                                     SV1  74Ø
   4Ø RDX1 = RDX2                                                   SV1  75Ø
      RDR1 = RDR2                                                   SV1  76Ø
      RDX2 = Ø.Ø                                                    SV1  77Ø
      RDR2 = Ø.Ø                                                    SV1  78Ø
      IF (K.EQ.Ø) GO TO 1ØØ                                         SV1  79Ø
C NEW RESIDUALS.                                                    SV1  8ØØ
      DO 5Ø I=1,M                                                   SV1  81Ø
        WRES(I) = WRES(I) + F(I)*W(I)                               SV1  82Ø
        IF (W(I).EQ.Ø.Ø) GO TO 5Ø                                   SV1  83Ø
        RES(I) = RES(I) + F(I)/W(I)                                 SV1  84Ø
   5Ø CONTINUE                                                      SV1  85Ø
      DO 7Ø NS=1,N1                                                 SV1  86Ø
        J = IPIVOT(NS)                                              SV1  87Ø
        X(J) = X(J) + G(NS)                                         SV1  88Ø
        SUM = Ø.Ø                                                   SV1  89Ø
        DO 6Ø L=1,M                                                 SV1  9ØØ
          SUM = SUM + DBLE(A(L,J))*DBLE(WRES(L))                    SV1  91Ø
   6Ø   CONTINUE                                                    SV1  92Ø
        G(NS) = -SUM                                                SV1  93Ø
   7Ø CONTINUE                                                      SV1  94Ø
      DO 9Ø I=1,M                                                   SV1  95Ø
        SUM = Ø.Ø                                                   SV1  96Ø
        IF (I.GT.M1) SUM = DBLE(RES(I))                             SV1  97Ø
        DO 8Ø L=1,N                                                 SV1  98Ø
          SUM = SUM + DBLE(A(I,L))*DBLE(X(L))                       SV1  99Ø
   8Ø   CONTINUE                                                    SV1 1ØØØ
        SUM = SUM - DBLE(B(I))                                      SV1 1Ø1Ø
        F(I) = -SUM                                                 SV1 1Ø2Ø
        F(I) = F(I)*W(I)                                            SV1 1Ø3Ø
        IF (W(I).EQ.Ø.Ø) RES(I) = DBLE(RES(I)) - SUM                SV1 1Ø4Ø
   9Ø CONTINUE                                                      SV1 1Ø5Ø
C END NEW RESIDUALS.                                                SV1 1Ø6Ø
  1ØØ MV = 1                                                        SV1 1Ø7Ø
      MH = M1                                                       SV1 1Ø8Ø
      DO 16Ø NS=1,N1                                                SV1 1Ø9Ø
        J = IPIVOT(NS)                                              SV1 11ØØ
        IF (NS.NE.M1+1) GO TO 11Ø                                   SV1 111Ø
        MV = M1 + 1                                                 SV1 112Ø
        MH = M                                                      SV1 113Ø
  11Ø   NSM1 = NS - 1                                               SV1 114Ø
        SUM = -DBLE(G(NS))                                          SV1 115Ø
        IF (1.GT.NSM1) GO TO 13Ø                                    SV1 116Ø
        DO 12Ø L=1,NSM1                                             SV1 117Ø
          SUM = SUM + DBLE(R(L,NS))*DBLE(Y(L))                      SV1 118Ø
  12Ø   CONTINUE                                                    SV1 119Ø
  13Ø   Y(NS) = -SUM                                                SV1 12ØØ
        C = Ø.Ø                                                     SV1 121Ø
        IF (NS.GT.M1) C = -Y(NS)                                    SV1 122Ø
        SUM = DBLE(C)                                               SV1 123Ø
        DO 14Ø L=MV,MH                                              SV1 124Ø
```

```
            SUM = SUM + DBLE(Q(L,J))*DBLE(F(L))                          SV1 125Ø
   14Ø      CONTINUE                                                     SV1 126Ø
            C = SUM                                                      SV1 127Ø
            C = C/D(NS)                                                  SV1 128Ø
            G(NS) = C                                                    SV1 129Ø
            DO 15Ø I=MV,M                                                SV1 13ØØ
              F(I) = F(I) - C*Q(I,J)                                     SV1 131Ø
   15Ø      CONTINUE                                                     SV1 132Ø
   16Ø CONTINUE                                                         SV1 133Ø
           IF (1.GT.M1) GO TO 21Ø                                       SV1 134Ø
           DO 17Ø I=1,M1                                                SV1 135Ø
             F(I) = Ø.Ø                                                 SV1 136Ø
   17Ø CONTINUE                                                         SV1 137Ø
           DO 2ØØ NS=1,M1                                               SV1 138Ø
             J = IPIVOT(NS)                                             SV1 139Ø
             SUM = -DBLE(Y(NS))                                         SV1 14ØØ
             DO 18Ø L=1,M                                               SV1 141Ø
               SUM = SUM + DBLE(Q(L,J))*DBLE(F(L))                      SV1 142Ø
   18Ø       CONTINUE                                                   SV1 143Ø
             C = SUM                                                    SV1 144Ø
             C = C/D(NS)                                                SV1 145Ø
             DO 19Ø I=1,M1                                              SV1 146Ø
               F(I) = F(I) - C*Q(I,J)                                   SV1 147Ø
   19Ø       CONTINUE                                                   SV1 148Ø
   2ØØ CONTINUE                                                         SV1 149Ø
   21Ø DO 24Ø I=1,N1                                                    SV1 15ØØ
             NS = N1 + 1 - I                                            SV1 151Ø
             NSP1 = NS + 1                                              SV1 152Ø
             SUM = -DBLE(G(NS))                                         SV1 153Ø
             IF (NSP1.GT.N1) GO TO 23Ø                                  SV1 154Ø
             DO 22Ø L=NSP1,N1                                           SV1 155Ø
               SUM = SUM + DBLE(R(NS,L))*DBLE(G(L))                     SV1 156Ø
   22Ø       CONTINUE                                                   SV1 157Ø
   23Ø       G(NS) = -SUM                                               SV1 158Ø
   24Ø CONTINUE                                                         SV1 159Ø
           DO 25Ø NS=1,N1                                               SV1 16ØØ
             RDX2 = RDX2 + G(NS)*G(NS)                                  SV1 161Ø
   25Ø CONTINUE                                                         SV1 162Ø
           DO 26Ø I=1,M                                                 SV1 163Ø
             RDR2 = RDR2 + F(I)*F(I)                                    SV1 164Ø
   26Ø CONTINUE                                                         SV1 165Ø
           IF (K.NE.Ø) GO TO 27Ø                                        SV1 166Ø
           RNX = RDX2                                                   SV1 167Ø
           RNR = RDR2                                                   SV1 168Ø
   27Ø IF (K.NE.1) GO TO 29Ø                                            SV1 169Ø
           XNORM = SQRT(RNX)                                            SV1 17ØØ
           DXNORM = SQRT(RDX2)                                          SV1 171Ø
           IF (XNORM.NE.Ø.Ø) GO TO 28Ø                                  SV1 172Ø
           DIGITX = -ALOG1Ø(ETA)                                        SV1 173Ø
           GO TO 29Ø                                                    SV1 174Ø
   28Ø DIGITX = -ALOG1Ø(AMAX1(DXNORM/XNORM,ETA))                        SV1 175Ø
C END K-TH ITERATION STEP.                                              SV1 176Ø
   29Ø NUMIT = K                                                        SV1 177Ø
           K = K + 1                                                    SV1 178Ø
           GO TO 3Ø                                                     SV1 179Ø
   3ØØ IF ((M1+KZ.EQ.M) .AND. (RDX2.GT.4.*ETA2*RNX)) GO TO 31Ø          SV1 18ØØ
           IF ((RDR2.GT.4.*ETA2*RNR) .AND.                              SV1 181Ø
          * (RDX2.GT.4.*ETA2*RNX)) GO TO 31Ø                            SV1 182Ø
           FAIL = .FALSE.                                               SV1 183Ø
           RETURN                                                       SV1 184Ø
   31Ø FAIL = .TRUE.                                                    SV1 185Ø
           RETURN                                                       SV1 186Ø
           END                                                          SV1 187Ø


C     SUBROUTINE SOLVE2(...)                                            SV2   1Ø
C SUBROUTINE SOLVE2 USES THE ORTHOGONAL DECOMPOSITION STORED IN QR, D    SV2   2Ø
C AND IPIVOT TO COMPUTE THE SOLUTION (COEFFICIENTS AND RESIDUALS)        SV2   3Ø
C TO THE LEAST SQUARES PROBLEM WHOSE RIGHT-HAND SIDE IS GIVEN IN B.      SV2   4Ø
C IN THE EVENT THAT N1 (THE COMPUTED RANK OF MATRIX H) IS LESS THAN N    SV2   5Ø
C (THE NUMBER OF UNKNOWN COEFFICIENTS), A UNIQUE SOLUTION VECTOR HAVING SV2   6Ø
C N ELEMENTS CAN BE OBTAINED BY IMPOSING THE CONDITION THAT THE          SV2   7Ø
C SOLUTION BE OF MINIMAL EUCLIDEAN NORM.  SUCH A SOLUTION IS SOUGHT IN  SV2   8Ø
C THE CASE OF UNDERDETERMINED OR RANK-DEFICIENT PROBLEMS.                SV2   9Ø
C IN NORMAL EXITS, THE SOLUTION IS CONTAINED IN THE VECTOR X             SV2  1ØØ
C (COEFFICIENTS) AND THE VECTOR RES (RESIDUALS).                         SV2  11Ø
```

```
C ITERATIVE REFINEMENT IS USED TO IMPROVE THE ACCURACY OF THE INITIAL   SV2  12Ø
C SOLUTION.                                                             SV2  13Ø
C ON EXIT, FAIL IS SET EQUAL TO .TRUE. IF THE SOLUTION FAILS TO         SV2  14Ø
C IMPROVE SUFFICIENTLY.  OTHERWISE, FAIL = .FALSE.  INFORMATION ON THE  SV2  15Ø
C BEHAVIOR OF THE ITERATIVE REFINEMENT PROCEDURE IS GIVEN BY NUMIT AND  SV2  16Ø
C DIGITX.  NUMIT IS THE NUMBER OF ITERATIONS CARRIED OUT IN ATTEMPTING  SV2  17Ø
C TO OBTAIN A SOLUTION.  DIGITX IS AN ESTIMATE OF THE NUMBER OF         SV2  18Ø
C CORRECT DIGITS IN THE INITIAL SOLUTION OF THE COEFFICIENTS.           SV2  19Ø
C THIS SUBROUTINE CALLS SUBROUTINE SOLVE3.                              SV2  2ØØ
C                                                                       SV2  21Ø
C ********* CONVERSION OF THIS SUBROUTINE TO DOUBLE PRECISION ********* SV2  22Ø
C * IF THE PROGRAM IS CONVERTED SO THAT ALL CALCULATIONS ARE DONE IN  * SV2  23Ø
C * DOUBLE PRECISION ARITHMETIC, THE ITERATIVE REFINEMENT PRESENTLY   * SV2  24Ø
C * INCLUDED IN SOLVE2 SHOULD BE OMITTED, SINCE THE SUCCESS OF THIS   * SV2  25Ø
C * PROCEDURE DEPENDS ON COMPUTING INNER PRODUCTS IN GREATER          * SV2  26Ø
C * PRECISION THAN OTHER CALCULATIONS.                               * SV2  27Ø
C * SEE COMMENTS IN SUBROUTINE L2B REGARDING CONVERSION TO DOUBLE     * SV2  28Ø
C * PRECISION.  IN ADDITION, THE FOLLOWING COMMENTS INDICATE HOW TO   * SV2  29Ø
C * OMIT THE ITERATIVE REFINEMENT FROM THIS SUBROUTINE.  STATEMENT    * SV2  3ØØ
C * NUMBERS GIVEN HERE REFER TO THOSE IN THE RIGHT-HAND MARGIN.       * SV2  31Ø
C *                                                                   * SV2  32Ø
C * 1. IN STATEMENT SV2 47Ø CHANGE REAL TO DOUBLE PRECISION.          * SV2  33Ø
C * 2. REPLACE STATEMENT SV2 88Ø BY A STATEMENT READING               * SV2  34Ø
C *       3Ø DO 5Ø I=1,M                                              * SV2  35Ø
C * 3. REPLACE STATEMENTS SV2 131Ø-14ØØ BY A STATEMENT READING        * SV2  36Ø
C *          RETURN                                                   * SV2  37Ø
C * 4. OMIT STATEMENTS SV2 12Ø-19Ø, 44Ø, 49Ø-5ØØ, 52Ø-55Ø, 65Ø,       * SV2  38Ø
C *    75Ø-85Ø, 165Ø-183Ø AND 185Ø-191Ø.                              * SV2  39Ø
C *                                                                   * SV2  4ØØ
C ********************************************************************* SV2  41Ø
C                                                                       SV2  42Ø
      SUBROUTINE SOLVE2(M, N, M1, A, B, W, N1, IPIVOT, QR, D,           SV2  43Ø
     * ETA, FAIL, NUMIT, DIGITX,                                        SV2  44Ø
     * X, RES, WRES, Y1, Y2, Y, F, G, MM, MMPNN)                        SV2  45Ø
      INTEGER IPIVOT(N)                                                 SV2  46Ø
      REAL A(MM,N), B(1), C, D(1), F(1), G(1),                          SV2  47Ø
     * QR(MMPNN,N), RES(1), W(M), WRES(1), X(1), Y(1), Y1(1), Y2(1)     SV2  48Ø
      REAL DIGITX, DXNORM, ETA, ETA2, RDR1, RDR2, RDX1, RDX2, RNR,      SV2  49Ø
     * RNX, XNORM                                                       SV2  5ØØ
      DOUBLE PRECISION SUM                                              SV2  51Ø
      LOGICAL FAIL                                                      SV2  52Ø
      NUMIT = Ø                                                         SV2  53Ø
      KZ = Ø                                                            SV2  54Ø
      ETA2 = ETA*ETA                                                    SV2  55Ø
      MP1 = M + 1                                                       SV2  56Ø
      MPN = M + N                                                       SV2  57Ø
      N1P1 = N1 + 1                                                     SV2  58Ø
      DO 1Ø I=1,M                                                       SV2  59Ø
        F(I) = B(I)*W(I)                                                SV2  6ØØ
        G(I) = Ø.Ø                                                      SV2  61Ø
        WRES(I) = Ø.Ø                                                   SV2  62Ø
        RES(I) = Ø.Ø                                                    SV2  63Ø
        Y1(I) = Ø.Ø                                                     SV2  64Ø
        IF (W(I).EQ.Ø.Ø) KZ = KZ + 1                                    SV2  65Ø
   1Ø CONTINUE                                                          SV2  66Ø
      DO 2Ø NS=1,N                                                      SV2  67Ø
        J = M + NS                                                      SV2  68Ø
        F(J) = Ø.Ø                                                      SV2  69Ø
        G(J) = Ø.Ø                                                      SV2  7ØØ
        X(NS) = Ø.Ø                                                     SV2  71Ø
        Y2(NS) = Ø.Ø                                                    SV2  72Ø
   2Ø CONTINUE                                                          SV2  73Ø
      K = Ø                                                             SV2  74Ø
      RDX2 = Ø.Ø                                                        SV2  75Ø
      RDR2 = Ø.Ø                                                        SV2  76Ø
C BEGIN K-TH ITERATION STEP.                                            SV2  77Ø
   3Ø IF (K.LT.2) GO TO 4Ø                                              SV2  78Ø
      IF (((64.*RDX2.LT.RDX1) .AND. (RDX2.GT.ETA2*RNX)) .OR.            SV2  79Ø
     * ((64.*RDR2.LT.RDR1) .AND. (RDR2.GT.ETA2*RNR))) GO TO 4Ø         SV2  8ØØ
      GO TO 27Ø                                                         SV2  81Ø
   4Ø RDX1 = RDX2                                                       SV2  82Ø
      RDR1 = RDR2                                                       SV2  83Ø
      RDX2 = Ø.Ø                                                        SV2  84Ø
      RDR2 = Ø.Ø                                                        SV2  85Ø
      IF (K.EQ.Ø) GO TO 16Ø                                             SV2  86Ø
C NEW RESIDUALS.                                                        SV2  87Ø
```

```
      DO 5Ø I=1,M                                                    SV2  88Ø
        WRES(I) = WRES(I) + F(I)*W(I)                                SV2  89Ø
        IF (W(I).EQ.Ø.Ø) GO TO 5Ø                                    SV2  9ØØ
        RES(I) = RES(I) + F(I)/W(I)                                  SV2  91Ø
        Y1(I) = Y1(I) + G(I)                                         SV2  92Ø
   5Ø CONTINUE                                                       SV2  93Ø
      DO 1ØØ NS=1,N                                                  SV2  94Ø
        J = M + NS                                                   SV2  95Ø
        NP = IPIVOT(NS)                                              SV2  96Ø
        X(NP) = X(NP) + F(J)                                         SV2  97Ø
        Y2(NP) = Y2(NP) + G(J)                                       SV2  98Ø
        SUM = -DBLE(X(NP))                                           SV2  99Ø
        DO 6Ø L=1,M                                                  SV2 1ØØØ
          SUM = SUM + DBLE(A(L,NP))*DBLE(Y1(L))                      SV2 1Ø1Ø
   6Ø   CONTINUE                                                     SV2 1Ø2Ø
        G(J) = -SUM                                                  SV2 1Ø3Ø
        IF (NS.GT.N1) GO TO 7Ø                                       SV2 1Ø4Ø
        GO TO 8Ø                                                     SV2 1Ø5Ø
   7Ø   F(J) = Ø.Ø                                                   SV2 1Ø6Ø
        GO TO 1ØØ                                                    SV2 1Ø7Ø
   8Ø   SUM = Ø.Ø                                                    SV2 1Ø8Ø
        DO 9Ø L=1,M                                                  SV2 1Ø9Ø
          SUM = SUM + DBLE(A(L,NP))*DBLE(WRES(L))                    SV2 11ØØ
   9Ø   CONTINUE                                                     SV2 111Ø
        F(J) = -SUM                                                  SV2 112Ø
  1ØØ CONTINUE                                                       SV2 113Ø
      DO 13Ø I=1,M                                                   SV2 114Ø
        SUM = Ø.Ø                                                    SV2 115Ø
        IF (I.GT.M1) SUM = DBLE(RES(I))                              SV2 116Ø
        DO 11Ø L=1,N                                                 SV2 117Ø
          SUM = SUM + DBLE(A(I,L))*DBLE(X(L))                        SV2 118Ø
  11Ø   CONTINUE                                                     SV2 119Ø
        SUM = SUM - DBLE(B(I))                                       SV2 12ØØ
        F(I) = -SUM                                                  SV2 121Ø
        F(I) = F(I)*W(I)                                             SV2 122Ø
        IF (W(I).EQ.Ø.Ø) RES(I) = DBLE(RES(I)) - SUM                 SV2 123Ø
        SUM = Ø.Ø                                                    SV2 124Ø
        IF (I.GT.M1) SUM = DBLE(Y1(I))                               SV2 125Ø
        DO 12Ø L=1,N                                                 SV2 126Ø
          SUM = SUM + DBLE(A(I,L))*DBLE(Y2(L))                       SV2 127Ø
  12Ø   CONTINUE                                                     SV2 128Ø
        G(I) = -SUM                                                  SV2 129Ø
  13Ø CONTINUE                                                       SV2 13ØØ
      IF (N1P1.GT.N) GO TO 16Ø                                       SV2 131Ø
      DO 15Ø I=N1P1,N                                                SV2 132Ø
        NS = N + N1P1 - I                                            SV2 133Ø
        J = M + NS                                                   SV2 134Ø
        SUM = Ø.Ø                                                    SV2 135Ø
        DO 14Ø L=1,J                                                 SV2 136Ø
          SUM = SUM + DBLE(QR(L,NS))*DBLE(G(L))                      SV2 137Ø
  14Ø   CONTINUE                                                     SV2 138Ø
        G(J) = SUM                                                   SV2 139Ø
  15Ø CONTINUE                                                       SV2 14ØØ
C END NEW RESIDUALS.                                                 SV2 141Ø
C                                                                    SV2 142Ø
  16Ø CALL SOLVE3(F, M1, M, N1, QR, D, Y, MMPNN)                     SV2 143Ø
C                                                                    SV2 144Ø
      IF (N1P1.GT.N) GO TO 2ØØ                                       SV2 145Ø
      DO 19Ø NS=N1P1,N                                               SV2 146Ø
        J = M + NS                                                   SV2 147Ø
        SUM = DBLE(G(J))                                             SV2 148Ø
        DO 17Ø L=MP1,J                                               SV2 149Ø
          SUM = SUM + DBLE(QR(L,NS))*DBLE(F(L))                      SV2 15ØØ
  17Ø   CONTINUE                                                     SV2 151Ø
        C = SUM                                                      SV2 152Ø
        C = C/D(NS)                                                  SV2 153Ø
        DO 18Ø I=1,J                                                 SV2 154Ø
          F(I) = F(I) - C*QR(I,NS)                                   SV2 155Ø
  18Ø   CONTINUE                                                     SV2 156Ø
  19Ø CONTINUE                                                       SV2 157Ø
  2ØØ DO 21Ø J=MP1,MPN                                               SV2 158Ø
        G(J) = Ø.Ø                                                   SV2 159Ø
        IF (J.LE.M+N1) G(J) = G(J) + F(J)                            SV2 16ØØ
  21Ø CONTINUE                                                       SV2 161Ø
C                                                                    SV2 162Ø
      CALL SOLVE3(G, M1, M, N1, QR, D, Y, MMPNN)                     SV2 163Ø
```

```
C                                                                       SV2 164Ø
      DO 22Ø I=1,M                                                      SV2 165Ø
        RDR2 = RDR2 + F(I)*F(I)                                         SV2 166Ø
  22Ø CONTINUE                                                          SV2 167Ø
      DO 23Ø I=MP1,MPN                                                  SV2 168Ø
        RDX2 = RDX2 + F(I)*F(I)                                         SV2 169Ø
  23Ø CONTINUE                                                          SV2 17ØØ
      IF (K.NE.Ø) GO TO 24Ø                                             SV2 171Ø
      RNR = RDR2                                                        SV2 172Ø
      RNX = RDX2                                                        SV2 173Ø
  24Ø IF (K.NE.1) GO TO 26Ø                                             SV2 174Ø
      XNORM = SQRT(RNX)                                                 SV2 175Ø
      DXNORM = SQRT(RDX2)                                               SV2 176Ø
      IF (XNORM.NE.Ø.Ø) GO TO 25Ø                                       SV2 177Ø
      DIGITX = -ALOG1Ø(ETA)                                             SV2 178Ø
      GO TO 26Ø                                                         SV2 179Ø
  25Ø DIGITX = -ALOG1Ø(AMAX1(DXNORM/XNORM,ETA))                         SV2 18ØØ
C END K-TH ITERATION STEP.                                              SV2 181Ø
  26Ø NUMIT = K                                                         SV2 182Ø
      K = K + 1                                                         SV2 183Ø
      GO TO 3Ø                                                          SV2 184Ø
  27Ø IF ((M1+KZ.EQ.M) .AND. (RDX2.GT.4.*ETA2*RNX)) GO TO 28Ø            SV2 185Ø
      IF ((RDR2.GT.4.*ETA2*RNR) .AND.                                   SV2 186Ø
     * (RDX2.GT.4.*ETA2*RNX)) GO TO 28Ø                                 SV2 187Ø
      FAIL = .FALSE.                                                    SV2 188Ø
      RETURN                                                            SV2 189Ø
  28Ø FAIL = .TRUE.                                                     SV2 19ØØ
      RETURN                                                            SV2 191Ø
      END                                                               SV2 192Ø


      SUBROUTINE SOLVE3(F, M1, M, N1, QR, D, Y, MMPNN)                  SV3   1Ø
C SUBROUTINE SOLVE3 IS CALLED ONLY BY SUBROUTINE SOLVE2.                SV3   2Ø
C THIS SUBROUTINE CALCULATES NEW VALUES OF F.                           SV3   3Ø
      REAL C, D(1), F(1), QR(MMPNN,N1), Y(1)                            SV3   4Ø
      DOUBLE PRECISION SUM                                              SV3   5Ø
      MV = 1                                                            SV3   6Ø
      MH = M1                                                           SV3   7Ø
      DO 1ØØ NS=1,N1                                                    SV3   8Ø
        J = M + NS                                                      SV3   9Ø
        IF (NS.EQ.M1+1) GO TO 1Ø                                        SV3  1ØØ
        GO TO 2Ø                                                        SV3  11Ø
   1Ø   MV = M1 + 1                                                     SV3  12Ø
        MH = M                                                          SV3  13Ø
   2Ø   NSM1 = NS - 1                                                   SV3  14Ø
        SUM = -DBLE(F(J))                                               SV3  15Ø
        IF (NS.EQ.1) GO TO 4Ø                                           SV3  16Ø
        DO 3Ø L=1,NSM1                                                  SV3  17Ø
          MPL = M + L                                                   SV3  18Ø
          SUM = SUM + DBLE(QR(MPL,NS))*DBLE(Y(L))                       SV3  19Ø
   3Ø   CONTINUE                                                        SV3  2ØØ
   4Ø   Y(NS) = -SUM                                                    SV3  21Ø
        IF (NS.GT.M1) GO TO 5Ø                                          SV3  22Ø
        GO TO 6Ø                                                        SV3  23Ø
   5Ø   C = -Y(NS)                                                      SV3  24Ø
        GO TO 7Ø                                                        SV3  25Ø
   6Ø   C = Ø.Ø                                                         SV3  26Ø
   7Ø   SUM = DBLE(C)                                                   SV3  27Ø
        DO 8Ø L=MV,MH                                                   SV3  28Ø
          SUM = SUM + DBLE(QR(L,NS))*DBLE(F(L))                         SV3  29Ø
   8Ø   CONTINUE                                                        SV3  3ØØ
        C = SUM                                                         SV3  31Ø
        C = C/D(NS)                                                     SV3  32Ø
        F(J) = C                                                        SV3  33Ø
        DO 9Ø L=MV,M                                                    SV3  34Ø
          F(L) = F(L) - C*QR(L,NS)                                      SV3  35Ø
   9Ø   CONTINUE                                                        SV3  36Ø
  1ØØ CONTINUE                                                          SV3  37Ø
      IF (1.GT.M1) GO TO 15Ø                                            SV3  38Ø
      DO 11Ø L=1,M1                                                     SV3  39Ø
        F(L) = Ø.Ø                                                      SV3  4ØØ
  11Ø CONTINUE                                                          SV3  41Ø
      DO 14Ø NS=1,M1                                                    SV3  42Ø
        SUM = -DBLE(Y(NS))                                              SV3  43Ø
        DO 12Ø L=1,M                                                    SV3  44Ø
          SUM = SUM + DBLE(QR(L,NS))*DBLE(F(L))                         SV3  45Ø
```

```
  12Ø   CONTINUE                                                    SV3  46Ø
        C = SUM                                                     SV3  47Ø
        C = C/D(NS)                                                 SV3  48Ø
        DO 13Ø L=1,M1                                               SV3  49Ø
          F(L) = F(L) - C*QR(L,NS)                                  SV3  5ØØ
  13Ø   CONTINUE                                                    SV3  51Ø
  14Ø CONTINUE                                                      SV3  52Ø
  15Ø DO 17Ø NS=1,N1                                                SV3  53Ø
        J = M + N1 + 1 - NS                                         SV3  54Ø
        MPN1 = M + N1                                               SV3  55Ø
        SUM = Ø.Ø                                                   SV3  56Ø
        DO 16Ø L=J,MPN1                                             SV3  57Ø
          LMM = L - M                                               SV3  58Ø
          SUM = SUM + DBLE(QR(J,LMM))*DBLE(F(L))                    SV3  59Ø
  16Ø   CONTINUE                                                    SV3  6ØØ
        F(J) = -SUM                                                 SV3  61Ø
  17Ø CONTINUE                                                      SV3  62Ø
      RETURN                                                        SV3  63Ø
      END                                                           SV3  64Ø


      SUBROUTINE COVAR(N, M1, N1, IPIVOT, C, D, Z, NN)              COV   1Ø
C SUBROUTINE COVAR USES RESULTS FROM THE ORTHOGONAL DECOMPOSITION   COV   2Ø
C STORED IN C, D AND IPIVOT TO COMPUTE THE UNSCALED COVARIANCE MATRIX COV  3Ø
C OF THE LEAST SQUARES COEFFICIENTS.                                COV   4Ø
C ON ENTRY, THE FIRST N ROWS AND THE FIRST N COLUMNS OF C CONTAIN THE COV  5Ø
C UPPER TRIANGULAR MATRIX OBTAINED FROM THE DECOMPOSITION.  THIS INPUT COV 6Ø
C MATRIX IS DESTROYED IN SUBSEQUENT CALCULATIONS.                   COV   7Ø
C ON EXIT, THE LOWER TRIANGULAR PORTION OF THE SYMMETRIC UNSCALED   COV   8Ø
C COVARIANCE MATRIX IS CONTAINED IN                                 COV   9Ø
C     C(1,1)                                                        COV  1ØØ
C     C(2,1) C(2,2)                                                 COV  11Ø
C     . . .                                                         COV  12Ø
C     C(N,1) C(N,2) ... C(N,N)                                      COV  13Ø
C IF N1 IS LESS THAN N, ONE OR MORE COLUMNS OF THE MATRIX           COV  14Ø
C H = (SQRT(W))*A WERE REJECTED AS BEING LINEARLY DEPENDENT.  WHENEVER COV 15Ø
C THE K-TH COLUMN OF H WAS SO REJECTED, C(I,J) IS SET EQUAL TO ZERO, COV 16Ø
C FOR I = K OR J = K, I.GE.J.                                       COV  17Ø
      INTEGER IPIVOT(N)                                             COV  18Ø
      REAL C(NN,N), D(1), Z(1)                                      COV  19Ø
      DOUBLE PRECISION SUM                                          COV  2ØØ
      L = N1                                                        COV  21Ø
      IF (L.GT.M1) C(L,L) = 1.Ø/D(L)                                COV  22Ø
      IF (L.EQ.1) GO TO 6Ø                                          COV  23Ø
   1Ø J = L - 1                                                     COV  24Ø
      IF (J.GT.M1) C(J,J) = 1.Ø/D(J)                                COV  25Ø
      DO 2Ø K=L,N1                                                  COV  26Ø
        Z(K) = C(J,K)                                               COV  27Ø
   2Ø CONTINUE                                                      COV  28Ø
      I = N1                                                        COV  29Ø
      DO 4Ø KA=J,N1                                                 COV  3ØØ
        SUM = Ø.Ø                                                   COV  31Ø
        IF (I.EQ.J) SUM = DBLE(C(I,J))                              COV  32Ø
        DO 3Ø K=L,N1                                                COV  33Ø
          SUM = SUM - DBLE(Z(K))*DBLE(C(K,I))                       COV  34Ø
   3Ø   CONTINUE                                                    COV  35Ø
        C(I,J) = SUM                                                COV  36Ø
        I = I - 1                                                   COV  37Ø
   4Ø CONTINUE                                                      COV  38Ø
      DO 5Ø K=L,N1                                                  COV  39Ø
        C(J,K) = C(K,J)                                             COV  4ØØ
   5Ø CONTINUE                                                      COV  41Ø
      L = L - 1                                                     COV  42Ø
      IF (L.GT.1) GO TO 1Ø                                          COV  43Ø
   6Ø IF (N1.EQ.N) GO TO 9Ø                                         COV  44Ø
      N1P1 = N1 + 1                                                 COV  45Ø
      DO 8Ø I=1,N                                                   COV  46Ø
        DO 7Ø J=N1P1,N                                              COV  47Ø
          C(I,J) = Ø.Ø                                              COV  48Ø
   7Ø   CONTINUE                                                    COV  49Ø
   8Ø CONTINUE                                                      COV  5ØØ
C PERMUTE THE COLUMNS AND ROWS OF MATRIX C TO ACCOUNT FOR PIVOTING. COV  51Ø
   9Ø DO 12Ø I=1,N                                                  COV  52Ø
        DO 1ØØ J=1,N                                                COV  53Ø
          K = IPIVOT(J)                                             COV  54Ø
          Z(K) = C(I,J)                                             COV  55Ø
```

```
 1ØØ   CONTINUE                                                    COV  56Ø
       DO 11Ø J=1,N                                                COV  57Ø
         C(I,J) = Z(J)                                             COV  58Ø
 11Ø   CONTINUE                                                    COV  59Ø
 12Ø CONTINUE                                                      COV  6ØØ
     DO 15Ø I=1,N                                                  COV  61Ø
       DO 13Ø J=1,N                                                COV  62Ø
         K = IPIVOT(J)                                             COV  63Ø
         Z(K) = C(J,I)                                             COV  64Ø
 13Ø   CONTINUE                                                    COV  65Ø
       DO 14Ø J=I,N                                                COV  66Ø
         C(J,I) = Z(J)                                             COV  67Ø
 14Ø   CONTINUE                                                    COV  68Ø
 15Ø CONTINUE                                                      COV  69Ø
     RETURN                                                        COV  7ØØ
     END                                                           COV  71Ø


     SUBROUTINE ERROR(IFAULT, K, Z)                                ERR   1Ø
C SUBROUTINE ERROR PRINTS ERROR DIAGNOSTICS IN THE CASE OF ERROR   ERR   2Ø
C FAILURE.                                                         ERR   3Ø
C ALSO PRINTED ARE SOME INFORMATIVE DIAGNOSTICS RELATED TO THE     ERR   4Ø
C ITERATIVE REFINEMENT OF ILL-CONDITIONED SYSTEMS OF EQUATIONS AND TO ERR  5Ø
C ROUNDING ERROR PROBLEMS IN COMPUTING THE COVARIANCE MATRIX.      ERR   6Ø
C IN THE DATA STATEMENT BELOW, NW IS THE PRINTER DEVICE NUMBER.    ERR   7Ø
     REAL Z                                                        ERR   8Ø
     DATA NW /6/                                                   ERR   9Ø
     GO TO (1Ø,2Ø,3Ø,4Ø,5Ø,6Ø,7Ø,8Ø,9Ø,1ØØ,11Ø), IFAULT            ERR  1ØØ
  1Ø WRITE (NW,99999)                                              ERR  11Ø
     RETURN                                                        ERR  12Ø
  2Ø WRITE (NW,99998)                                              ERR  13Ø
     RETURN                                                        ERR  14Ø
  3Ø WRITE (NW,99997)                                              ERR  15Ø
     IF (K.EQ.2) WRITE (NW,99996)                                  ERR  16Ø
     RETURN                                                        ERR  17Ø
  4Ø WRITE (NW,99995) K,Z                                          ERR  18Ø
     RETURN                                                        ERR  19Ø
  5Ø WRITE (NW,99994)                                              ERR  2ØØ
     RETURN                                                        ERR  21Ø
  6Ø WRITE (NW,99993)                                              ERR  22Ø
     RETURN                                                        ERR  23Ø
  7Ø WRITE (NW,99992)                                              ERR  24Ø
     RETURN                                                        ERR  25Ø
  8Ø WRITE (NW,99991) K                                            ERR  26Ø
     RETURN                                                        ERR  27Ø
  9Ø WRITE (NW,9999Ø) K,Z                                          ERR  28Ø
     RETURN                                                        ERR  29Ø
 1ØØ WRITE (NW,99989) K,Z                                          ERR  3ØØ
     RETURN                                                        ERR  31Ø
 11Ø WRITE (NW,99988) K,Z                                          ERR  32Ø
     RETURN                                                        ERR  33Ø
C FORMAT STATEMENTS.                                               ERR  34Ø
99999 FORMAT (5ØHØ***  PARAMETER ERROR.  M, N AND L MUST BE GREATER, ERR  35Ø
    * 11H THAN ZERO.)                                              ERR  36Ø
99998 FORMAT (5ØHØ***  PARAMETER ERROR.  M1 CANNOT EXCEED M OR N, B, ERR  37Ø
    * 26HUT M1 MUST BE NONNEGATIVE.)                               ERR  38Ø
99997 FORMAT (5ØHØ***  DIMENSION ERROR.  ONE OR MORE OF THE FOLLOWI, ERR  39Ø
    * 32HNG ERROR CONDITIONS WAS FOUND --/7X,12HM EXCEEDS MM/7X,    ERR  4ØØ
    * 12HN EXCEEDS NN)                                             ERR  41Ø
99996 FORMAT (5X,17HM+N EXCEEDS MMPNN)                             ERR  42Ø
99995 FORMAT (43HØ***  WEIGHTS MUST BE NONNEGATIVE.  FOR I =,I3,2X, ERR  43Ø
    * 9HWEIGHT = ,G15.8)                                           ERR  44Ø
99994 FORMAT (5ØHØ***  EITHER MATRIX H EQUALS ZERO OR MATRIX OF CON, ERR  45Ø
    * 51HSTRAINTS EQUALS ZERO.  NO SOLUTION CAN BE COMPUTED.)      ERR  46Ø
99993 FORMAT (5ØHØ***  SINCE THE CONSTRAINTS ARE LINEARLY DEPENDENT, ERR  47Ø
    * 29H NO SOLUTION CAN BE COMPUTED.)                            ERR  48Ø
99992 FORMAT (39HØ***  ALL SOLUTIONS FAILED TO CONVERGE.)          ERR  49Ø
99991 FORMAT (22HØ***  FOR B-VECTOR NO.,I3,21H SOLUTION FAILED TO C, ERR  5ØØ
    * 8HONVERGE.)                                                  ERR  51Ø
9999Ø FORMAT (22HØ***  FOR B-VECTOR NO.,I3,21H THE NUMBER OF ITERAT, ERR  52Ø
    * 45HIONS REQUIRED FOR CONVERGENCE OF SOLUTION WAS,F4.Ø/4H ***, ERR  53Ø
    * 57H  THIS NUMBER IS LARGE, INDICATING THE PROBLEM IS ILL-CON, ERR  54Ø
    * 4ØHDITIONED AND SOLUTION MAY BE INACCURATE.)                  ERR  55Ø
99989 FORMAT (22HØ***  FOR B-VECTOR NO.,I3,21H ESTIMATED NUMBER OF , ERR  56Ø
    * 54HCORRECT DIGITS IN INITIAL SOLUTION OF COEFFICIENTS IS ,    ERR  57Ø
    * G15.8/51H ***  SINCE THIS IS SMALL, THE FINAL SOLUTION MAY B, ERR  58Ø
```

```
     * 13HE INACCURATE.)                                                  ERR  59Ø
99988 FORMAT (26HØ***  DIAGONAL ELEMENT NO.,I3,17H OF THE UNSCALED ,     ERR  6ØØ
     * 57HCOVARIANCE MATRIX WAS COMPUTED TO BE NEGATIVE OWING TO RO,     ERR  61Ø
     * 13HUNDING ERROR./29H ***  THE COMPUTED VALUE WAS ,G15.8)          ERR  62Ø
      END                                                                ERR  63Ø
    1     MODE 1    ************************************************************
(1) WAMPLER, J.AMER.STAT.ASSN. 197Ø, P.549, 5TH DEG. POLYNOMIALS, EQUAL WEIGHTS.
   21    6    Ø    2    1    1    1        Ø.
(F2.Ø,2F8.Ø)
 Ø       1     76Ø
 1       6   -2Ø42
 2      63    2111
 3     364   -1684
 4    1365    3888
 5    39Ø6    1858
 6    9331   11379
 7   196Ø8   1756Ø
 8   37449   39287
 9   6643Ø   64382
1Ø  111111  113159
11  177156  1751Ø8
12  271453  273291
13  4Ø2234  4ØØ186
14  579195  581243
15  813616  811568
16 1118481 1121ØØ4
17 15Ø8598 15Ø655Ø
18 2ØØØ719 2ØØ2767
19 261366Ø 2611612
2Ø 3368421 336918Ø
    1
(2) FIRST DEGREE POLYNOMIAL, UNEQUAL WEIGHTS.
    6    2    Ø    1    1    2    1        Ø.
(3F3.Ø)
 1. 2. 2.
 2. 2. 1.
 3. 5. 1.
 4. 4. 1.
 5. 7. 1.
 6. 7. 2.
    1
(3) J. M. CAMERON DATA, UNEQUAL WEIGHTS, TWO COLUMNS LINEARLY DEPENDENT.
    7    6    Ø    2    2    2    1        Ø.
(2F3.Ø,F3.1,F3.Ø,F4.2,F3.Ø,F5.1,F4.Ø,F2.Ø)
  1  1 .5  2 .25  2 13.Ø 13Ø 2
  1  2 .5  2 .25  3 17.Ø 17Ø 2
  Ø  3 .Ø  3 .ØØ  3 18.2 182 1
  Ø  2 .Ø  1 .ØØ  1  8.8  88 1
  Ø  1 .Ø -3 .ØØ  Ø -3.Ø -3Ø 1
  Ø  1 .Ø  Ø .ØØ  Ø  2.8  28 1
  Ø  Ø .Ø  1 .ØØ  Ø  2.1  21 1
    1
(4) EXAMPLE WITH WEIGHTS AND CONSTRAINTS.
   12    6    3    1    2    2    1        Ø.
(8F3.Ø)
  1  1  1  1  1  1  6  1
  1  1  1  Ø  Ø  Ø  3  1
  1  1  Ø  Ø  Ø  Ø  2  1
  1 -1  Ø  Ø  Ø  Ø  1  3
  1  Ø -1  Ø  Ø  Ø -1  3
  1  Ø  Ø -1  Ø  Ø  1  3
  1  Ø  Ø  Ø  Ø -1 -1  2
  Ø  1 -1  Ø  Ø  Ø  1  2
  Ø  1  Ø  Ø -1  Ø -1  2
  Ø  1  Ø  Ø  Ø -1  1  1
  Ø  Ø  1 -1  Ø  Ø -1  1
  Ø  Ø  1  Ø -1  Ø  1  1
    1
(5) INVERSE OF HILBERT MATRIX OF ORDER 4.  M = 4, N = 4, M1 = Ø.
    4    4    Ø    1    2    1    1        Ø.
  (5F7.Ø)
    16.  -12Ø.   24Ø.  -14Ø.    -4.
  -12Ø.  12ØØ. -27ØØ.  168Ø.    6Ø.
   24Ø. -27ØØ.  648Ø. -42ØØ.  -18Ø.
  -14Ø.  168Ø. -42ØØ.  28ØØ.   14Ø.
    1
```

```
(6) INVERSE OF HILBERT MATRIX OF ORDER 4.  M = 4, N = 4, M1 = 4.
   4    4    4    1    2    1    1          Ø.
 (5F7.Ø)
   16.  -12Ø.   24Ø.  -14Ø.    -4.
 -12Ø.  12ØØ. -27ØØ.  168Ø.    6Ø.
  24Ø. -27ØØ.  648Ø. -42ØØ.  -18Ø.
 -14Ø.  168Ø. -42ØØ.  28ØØ.   14Ø.
   1
(7) BUSINGER-GOLUB, NUM. MATH. 1965, P.269, INVERSE OF HILBERT MATRIX, ORDER 6.
   6    5    1    2    2    1    1          Ø.
(5F1Ø.Ø,1ØX,2F1Ø.Ø)
       36.      -63Ø.      336Ø.     -756Ø.      756Ø.                   463.       463.
     -63Ø.     147ØØ.    -882ØØ.    21168Ø.   -22Ø5ØØ.                -1386Ø.    -1782Ø.
     336Ø.    -882ØØ.    56448Ø.  -14112ØØ.   15120ØØ.                 97Ø2Ø.     93555.
    -756Ø.    21168Ø.  -14112ØØ.   36288ØØ.  -39690ØØ.               -25872Ø.   -2618ØØ.
     756Ø.   -22Ø5ØØ.   15120ØØ.  -39690ØØ.   44100ØØ.                29106Ø.    288288.
    -2772.     8316Ø.   -58212Ø.   155232Ø.  -174636Ø.               -116424.   -118944.
   1
(8) EXAMPLE WITH X = Ø (HENCE XNORM = Ø).  TOL = -1 ON ENTRY TO L2A OR L2B.
   1    1    Ø    1    2    1    Ø          -1.
(2F3.Ø)
 1. Ø.
   1
(9) ALBERT, REGRESSION AND THE MOORE-PENROSE INVERSE, 1972, P. 63.
   3    4    Ø    3    2    1    2          Ø.
(7F4.Ø)
 1.  Ø.  1.  1.  1.  Ø.  Ø.
 Ø.  1. -1.  Ø.  Ø.  1.  Ø.
 1.  1.  Ø.  1.  Ø.  Ø.  1.
   1
(1Ø) FIFTH DEGREE POLYNOMIAL WITH HEAVY WEIGHTS, MATRIX A SCALED.      IFAULT=11
  21    6    Ø    1    2    2    1          Ø.
(F8.Ø,3F9.Ø,2F1Ø.Ø,F13.2,F1Ø.Ø)
1ØØØØØØ.       Ø.       Ø.       Ø.        Ø.        Ø.  1ØØØØØ.1853 16777216.
1ØØØØØØ.  1ØØØØØ.   1ØØØØ.    1ØØØ.      1ØØ.        1.  -8277497.ØØ        1.
1ØØØØØØ.  2ØØØØØ.   4ØØØØ.    8ØØØ.     16ØØ.       32.   85136ØØ.ØØ        1.
1ØØØØØØ.  3ØØØØØ.   9ØØØØ.   27ØØØ.     81ØØ.      243.  -8245855.ØØ        1.
1ØØØØØØ.  4ØØØØØ.  16ØØØØ.   64ØØØ.    256ØØ.     1Ø24.  1Ø5ØØ192.ØØ        1.
1ØØØØØØ.  5ØØØØØ.  25ØØØØ.  125ØØØ.    625ØØ.     3125.  -8191733.ØØ        1.
1ØØØØØØ.  6ØØØØØ.  36ØØØØ.  216ØØØ.   1296ØØ.     7776.   8626944.ØØ        1.
1ØØØØØØ.  7ØØØØØ.  49ØØØØ.  343ØØØ.   24Ø1ØØ.    168Ø7.  -8Ø94491.ØØ        1.
1ØØØØØØ.  8ØØØØØ.  64ØØØØ.  512ØØØ.   4Ø96ØØ.    32768.   7897376.ØØ        1.
1ØØØØØØ.  9ØØØØØ.  81ØØØØ.  729ØØØ.   6561ØØ.    59Ø49.  -792ØØ49.ØØ        1.
1ØØØØØØ. 1ØØØØØØ. 1ØØØØØØ. 1ØØØØØØ.  1ØØØØØØ.   1ØØØØØ.    6ØØØØØ.5Ø 16777216.
1ØØØØØØ. 11ØØØØØ. 121ØØØØ. 1331ØØØ.  14641ØØ.   161Ø51.  -7617Ø47.ØØ        1.
1ØØØØØØ. 12ØØØØØ. 144ØØØØ. 1728ØØØ.  2Ø736ØØ.   248832.   852144Ø.ØØ        1.
1ØØØØØØ. 13ØØØØØ. 169ØØØØ. 2197ØØØ.  28561ØØ.   371293.  -7113ØØ5.ØØ        1.
1ØØØØØØ. 14ØØØØØ. 196ØØØØ. 2744ØØØ.  38416ØØ.   537824.  1ØØ2Ø992.ØØ        1.
1ØØØØØØ. 15ØØØØØ. 225ØØØØ. 3375ØØØ.  5Ø625ØØ.   759375.  -631Ø483.ØØ        1.
1ØØØØØØ. 16ØØØØØ. 256ØØØØ. 4Ø96ØØØ.  65536ØØ.  1Ø48576.  12963744.ØØ        1.
1ØØØØØØ. 17ØØØØØ. 289ØØØØ. 4913ØØØ.  83521ØØ.  1419857.  -5Ø83241.ØØ        1.
1ØØØØØØ. 18ØØØØØ. 324ØØØØ. 5832ØØØ. 1Ø4976ØØ.  1889568.  12515136.ØØ        1.
1ØØØØØØ. 19ØØØØØ. 361ØØØØ. 6859ØØØ. 13Ø321ØØ.  2476Ø99.  -3272399.ØØ        1.
1ØØØØØØ. 2ØØØØØØ. 4ØØØØØØ. 8ØØØØØØ. 16ØØØØØØ.  32ØØØØØ. 63ØØØØØ.1853 16777216.
   1
(11) LAWSON-HANSON, SOLVING LEAST SQUARES PROBLEMS, 1974, SET 1 EX.16. IFAULT=1Ø
   8    6    4    1    2    1    1          Ø.
(7F6.Ø)
  155.  1Ø5. -445. -495.  -45.  -95. -245.
  355.  3Ø5. -245. -295.  155.  1Ø5. -295.
 -445. -495.  -45.  -95.  355.  3Ø5.  155.
 -245. -295.  155.  1Ø5. -445. -495.  1Ø5.
  -45.  -95.  355.  3Ø5. -245. -295. -445.
  155.  1Ø5. -445. -495.  -45.  -95. -495.
  355.  3Ø5. -245. -295.  155.  1Ø5.  -45.
 -445.. -495.  -45.  -95.  355.  3Ø5.  -95.
   1
(12) LAWSON-HANSON, SOLVING LEAST SQUARES PROBLEMS, P.252, SET 1, EX.16, TOL=.5.
   8    6    4    1    2    1    2          Ø.5
(7F6.Ø)
  155.  1Ø5. -445. -495.  -45.  -95. -245.
  355.  3Ø5. -245. -295.  155.  1Ø5. -295.
 -445. -495.  -45.  -95.  355.  3Ø5.  155.
 -245. -295.  155.  1Ø5. -445. -495.  1Ø5.
  -45.  -95.  355.  3Ø5. -245. -295. -445.
  155.  1Ø5. -445. -495.  -45.  -95. -495.
```

```
  355.  3Ø5. -245. -295.  155.  1Ø5.  -45.
 -445. -495.  -45.  -95.  355.  3Ø5.  -95.
    1
(13) BJORCK-GOLUB, BIT 1967, P.322, HILBERT MATRIX INVERSE, ORDER 8.  IFAULT=8,9
    8    6    2    3    2    1    1        Ø.
(6F12.Ø)
      2Ø16Ø.     -924ØØ.     221760.    -288288.     192192.     -5148Ø.
        945.        945.    84ØØ945.
    -952560.    465696Ø.  -11642ØØ.   15567552.  -1Ø594584.    288288Ø.
     -4Ø32Ø.     -4Ø32Ø.    415968Ø.
   1143Ø72Ø.  -58212ØØØ.  149688ØØØ. -2Ø432412Ø.  141261120.  -3891888Ø.
     456120.    325612Ø.    325612Ø.
  -58212ØØØ.  3Ø492ØØØØ. -8ØØ415ØØØ. 11Ø99Ø88ØØ. -77693616Ø.  216216ØØØ.
   -2236Ø8Ø.    -136Ø8Ø.    -136Ø8Ø.
  149688ØØØ. -8ØØ415ØØØ. 2134440ØØØ.-299675376Ø. 2118916800. -594594ØØØ.
    559944Ø.    7279440.    7279440.
 -2Ø432412Ø. 11Ø99Ø88ØØ.-299675376Ø. 4249941696.-3Ø3ØØ51Ø24.  856215360.
   -7495488.   -6Ø95488.   -6Ø95488.
  141261120. -77693616Ø. 2118916800.-3Ø3ØØ51Ø24. 2175421248. -618377760.
    51Ø51ØØ.    63Ø51ØØ.    63Ø51ØØ.
  -3891888Ø.  216216ØØØ. -594594ØØØ.  856215360. -618377760.  176679360.
   -1389960.    -33996Ø.    -33996Ø.
    1
(14) LAWSON-HANSON, SOLVING LEAST SQUARES PROBLEMS, 1974, SET 1, EX.12. IFAULT=7
    6    8    6    1    2    1    1        Ø.
(9F6.Ø)
 -245. -295.  155.  1Ø5. -445. -495.  -45.  -95.  355.    1.
  355.  3Ø5. -245. -295.  155.  1Ø5. -445. -495.  3Ø5.    1.
  -45.  -95.  355.  3Ø5. -245. -295.  155.  1Ø5. -245.    1.
 -445. -495.  -45.  -95.  355.  3Ø5. -245. -295. -295.    4.
  155.  1Ø5. -445. -495.  -45.  -95.  355.  3Ø5.  155.    9.
 -245. -295.  155.  1Ø5. -445. -495.  -45.  -95.  1Ø5.   16.
    1
(15) EXAMPLE WITH SINGULAR MATRIX OF CONSTRAINTS.  M1 = 3, N1 = 2.      IFAULT=6
    6    3    3    1    2    1    1        Ø.
(4F2.Ø)
 1 1 1 1
 2 2 2 1
 1 Ø Ø 1
 1 2 4 1
 1 3 9 1
 1 4 9 1
    1
(16) EXAMPLE WITH MATRIX A EQUAL TO ZERO (HENCE RANK EQUALS ZERO).      IFAULT=5
    3    2    Ø    1    2    1    1        Ø.
(3F2.Ø)
 Ø Ø 1
 Ø Ø 1
 Ø Ø 1
    1
(17) EXAMPLE WITH ZERO AND NEGATIVE WEIGHTS.                            IFAULT=4
    2    1    Ø    1    2    2    1        Ø.
(2F3.Ø,F4.Ø)
 1. 1.  Ø.
 1. 1. -1.
    1
(18) EXAMPLE WHERE N EXCEEDS THE CORRESPONDING DIMENSION LIMIT.         IFAULT=3
    1   21    Ø    1    2    1    2        Ø.
   (22F2.Ø)
 1 2 3 4 5 6 7 8 9 Ø 1 2 3 4 5 6 7 8 9 Ø 1 1
    1
(19) EXAMPLE WHERE M1 EXCEEDS M AND N.                                  IFAULT=2
    1    1    2    1    2    1    1        Ø.
(2F3.Ø)
 1. 1.
    1
(2Ø) EXAMPLE WITH M = Ø.                                                IFAULT=1
    Ø    1    Ø    1    2    1    1        Ø.
(2F3.Ø)
 1. 1. 1.
    Ø
    2     MODE 2    ************************************************************
(3) J. M. CAMERON DATA, UNEQUAL WEIGHTS, TWO COLUMNS LINEARLY DEPENDENT.
    7    6    Ø    2    2    2    1        Ø.
(2F3.Ø,F3.1,F3.Ø,F4.2,F3.Ø,F5.1,F4.Ø,F2.Ø)
  1  1 .5  2 .25  2 13.Ø 13Ø 2
```

```
  1  2 .5  2 .25  3 17.Ø 17Ø 2
  Ø  3 .Ø  3 .ØØ  3 18.2 182 1
  Ø  2 .Ø  1 .ØØ  1  8.8  88 1
  Ø  1 .Ø -3 .ØØ  Ø -3.Ø -3Ø 1
  Ø  1 .Ø  Ø .ØØ  Ø  2.8  28 1
  Ø  Ø .Ø  1 .ØØ  Ø  2.1  21 1
    1
(9) ALBERT. REGRESSION AND THE MOORE-PENROSE INVERSE. 1972. P. 63.
    3    4    Ø    3    2    1    2        Ø.
(7F4.Ø)
  1.  Ø.  1.  1.  1.  Ø.  Ø.
  Ø.  1. -1.  Ø.  Ø.  1.  Ø.
  1.  1.  Ø.  1.  Ø.  Ø.  1.
    1
(12) LAWSON-HANSON, SOLVING LEAST SQUARES PROBLEMS, P.252, SET 1, EX.16, TOL=.5.
    8    6    4    1    2    1    2       Ø.5
(7F6.Ø)
  155.  1Ø5. -445. -495.  -45.  -95. -245.
  355.  3Ø5. -245. -295.  155.  1Ø5. -295.
 -445. -495.  -45.  -95.  355.  3Ø5.  155.
 -245. -295.  155.  1Ø5. -445. -495.  1Ø5.
  -45.  -95.  355.  3Ø5. -245. -295. -445.
  155.  1Ø5. -445. -495.  -45.  -95. -495.
  355.  3Ø5. -245. -295.  155.  1Ø5.  -45.
 -445. -495.  -45.  -95.  355.  3Ø5.  -95.
    1
(14) LAWSON-HANSON, SOLVING LEAST SQUARES PROBLEMS, 1974, SET 1, EX.12. IFAULT=7
    6    8    6    1    2    1    1        Ø.
(9F6.Ø)
 -245. -295.  155.  1Ø5. -445. -495.  -45.  -95.  355.    1.
  355.  3Ø5. -245. -295.  155.  1Ø5. -445. -495.  3Ø5.    1.
  -45.  -95.  355.  3Ø5. -245. -295.  155.  1Ø5. -245.    1.
 -445. -495.  -45.  -95.  355.  3Ø5. -245. -295. -295.    4.
  155.  1Ø5. -445. -495.  -45.  -95.  355.  3Ø5.  155.    9.
 -245. -295.  155.  1Ø5. -445. -495.  -45.  -95.  1Ø5.   16.
    1
(18) EXAMPLE WHERE N EXCEEDS THE CORRESPONDING DIMENSION LIMIT.         IFAULT=3
    1   21    Ø    1    2    1    2        Ø.
   (22F2.Ø)
 1 2 3 4 5 6 7 8 9 Ø 1 2 3 4 5 6 7 8 9 Ø 1 1
    Ø
    1    MODE 1    ************************************************************
(21) FIRST DEGREE POLYNOMIAL, POSITIVE AND ZERO WEIGHTS.  COMPARE EXAMPLE (2).
    8    2    Ø    1    1    2             Ø.
(3F3.Ø)
 1. 2. 2.
 2. 2. 1.
 3. 5. 1.
 4. 4. 1.
 5. 7. 1.
 6. 7. 2.
 7. 9. Ø.
 8. 6. Ø.
    1
(22) EXAMPLE WITH WEIGHTS AND CONSTRAINTS.  COMPARE EXAMPLE (4).
   14    6    3    1    2    2             Ø.
(8F3.Ø)
  1  1  1  1  1  1  6  1
  Ø  Ø  1  Ø -1  Ø  1  1
  1  Ø  Ø  Ø -1  Ø -1  Ø
  Ø  1  Ø -1  Ø  Ø  1  Ø
    1
(23) EXAMPLE WITH ZERO WEIGHTS, WHERE RANK A = RANK H = 2.  H = (SQRT(W))*A.
    4    2    Ø    1    2    2             Ø.
(4F3.Ø)
 1. Ø. 1. 1.
 Ø. 1. 2. 1.
 Ø. Ø. 3. Ø.
 Ø. Ø. 4. Ø.
    1
(24) EXAMPLE WITH ZERO WEIGHTS, WHERE RANK A = 2, RANK H = 1.  H = (SQRT(W))*A.
    4    2    Ø    1    2    2             Ø.
(4F3.Ø)
 1. Ø. 1. 1.
 Ø. 1. 2. Ø.
 Ø. Ø. 3. Ø.
```

```
Ø. Ø. 4. 1.
   Ø
1  1  1  Ø  Ø  Ø  3  1
1  1  Ø  Ø  Ø  Ø  2  1
1 -1  Ø  Ø  Ø  Ø  1  3
1  Ø -1  Ø  Ø  Ø -1  3
1  Ø  Ø -1  Ø  Ø  1  3
1  Ø  Ø  Ø  Ø -1 -1  2
Ø  1 -1  Ø  Ø  Ø  1  2
Ø  1  Ø  Ø -1  Ø -1  2
Ø  1  Ø  Ø  Ø -1  1  1
Ø  Ø  1 -1  Ø  Ø -1  1
```

ALGORITHM 545
An Optimized Mass Storage FFT [C6]

DONALD FRASER

CSIRO, Australia

Key Words and Phrases: multidimensional FFT, fast Fourier transform, FFT, mass storage FFT, mass store sorting, optimal sorting
CR Categories: 4.9, 5.19, 5.31
Language: Fortran

DESCRIPTION

The program is an implementation of the optimal sorting algorithm of the author [8] which allows a base-2 version of the Cooley–Tukey FFT algorithm [2–4] efficient access to a mass store array. Optimal sorting for the mass storage FFT has been determined independently by DeLotto and Dotti [5, 6], but in the author's version the emphasis is on "in-place" array modification. This results in slightly higher mass store I/O than the minimum, but requires no additional mass store working space. The method is a logical extension of the work of Singleton [9] and Brenner [1].

The program computes in place the discrete Fourier transform of a one-dimensional or a multidimensional array. In the one-dimensional case the transform is defined by

$$A(J) = SCAL \sum_{j=0}^{N1-1} a(j) \exp(\pm i\, 2\pi jJ/N1) \quad \text{for} \quad J = 0, 1, \ldots, N1 - 1 \quad (1)$$

where SCAL is an arbitrary scaling factor, exp is the exponential function, and $i = \sqrt{-1}$, the sign of the exponent being either plus or minus, depending on the desired transform.

The definition (1) is easily generalized to cover more than one dimension; for example, the two-dimensional case is given by

$$A(K, J) = SCAL \sum_{k=0}^{N2-1} \sum_{j=0}^{N1-1} a(j, k) \exp(\pm i\, 2\pi(jJ/N1 + kK/N2))$$

$$\text{for} \quad K = 0, 1, \ldots, N2 - 1 \quad \text{and} \quad J = 0, 1, \ldots, N1 - 1. \quad (2)$$

The elements a(j) or a(j, k) in (1) or (2) are the initial complex data in a mass store array. These are replaced by the elements A(J) or A(K, J) as the final complex data. Index reversal in the two-dimensional case is used to indicate dimension transposition by the program (dimension order reversal in general). This may be suppressed in certain cases.

The program consists of a set of subroutines written in ANSI Fortran, only two

Received 24 October 1975 and 27 June 1979.

Author's address: Division of Computing Research, CSIRO, P.O. Box 1800, Canberra City, ACT 2601, Australia.

of which are called by a user program. If the data are always complex, the user program calls subroutine CMFFT. If the input data or the result are real, the user program may call the faster subroutine RMFFT. Either subroutine replaces a one-dimensional or a multidimensional array in mass store by its discrete Fourier transform, as defined by eq. (1) or (2).

(If the elements of the mass store array are considered to be singly indexed by i = 0, 1, 2, . . . , then the indices of the definitions map into i = j in the one-dimensional case or i = j + k·N1 in the two-dimensional case, resultant transposition giving i = K + J·N2. In general, i = j1 + j2·N1 + j3·N2·N1 + ... Remember also that indices are increased by 1 for Fortran.)

The mass store file is assumed to exist and to have been previously defined to the Fortran system and opened for random access by the user. The file is accessed through two system-dependent subroutines MFREAD and MFWRIT (see Appendix F). All other subroutines are system independent.

The user has freedom to specify total array size, mass store block or record size, core store working space size, and the dimensioning of the array, except that all sizes are to base 2 and are given by their binary exponents.

Complex data are transformed by an in-place, base-2 algorithm using postcomputation bit reversal to sort the array [2]. The computation is handled by a modified, in-core FFT routine which does a sequence of partial transforms of the mass store array (the method is discussed further in the following section). The sorting algorithm [8] calculates the most efficient way to access the mass store array for these computations. Finally, the sorting algorithm is used to carry out an overall bit-reversed permutation of the array, again with as few accesses as possible (I/O efficiency is discussed in the conclusion section).

Computation and sorting occur in place, through a combination of "virtual" permutations, where mass store blocks are accessed according to an indexing algorithm but are left physically unpermuted, and symmetric permutations, which interchange blocks according to a generalized index bit reversal. In [8] it is shown that any unsymmetric permutation can be formed from two suitable symmetric permutations, each of which can be done in place.

Multidimensional transforms are achieved automatically by making use of the indexing structure of the FFT algorithm itself. No change to the order of access of elements is necessary, so that the full advantage of the sorting algorithm and program simplicity is maintained.

For transforms in which the initial or final data are real, the usual time-saving algorithm [4] is available to unscramble a half-length complex transform of packed real data (or vice versa). This requires an extra accessing and computing pass through the mass store array but still results in a saving of nearly half the computation and I/O time. In this method it is easy to allow a choice in the degree of redundancy in the final, complex result. The array may be expanded to full redundancy (twice the physical length of the original packed real array), or to partial redundancy, or maintain the same physical length by elimination of all redundancy.

Finally, we must define a number of terms used in the discussion. "Core store" is used to describe a region where the elements of an array are equally accessible at random. "Mass store" implies a region where elements are grouped into "blocks" or "records" which must be accessed as units, but which units are accessible efficiently at random. A "pass" is an array operation which leaves the array elements in place ready for another array operation. Thus an "I/O pass" reads and writes back once all the blocks of an array (this is a logical definition—sometimes in practice not all blocks need be physically accessed, or some may be accessed more than once).

Algorithm Details

General. The structure of the mass storage FFT program is shown in Figure 1. When using real data, routine RMFFT calls CMFFT with a half-length complex

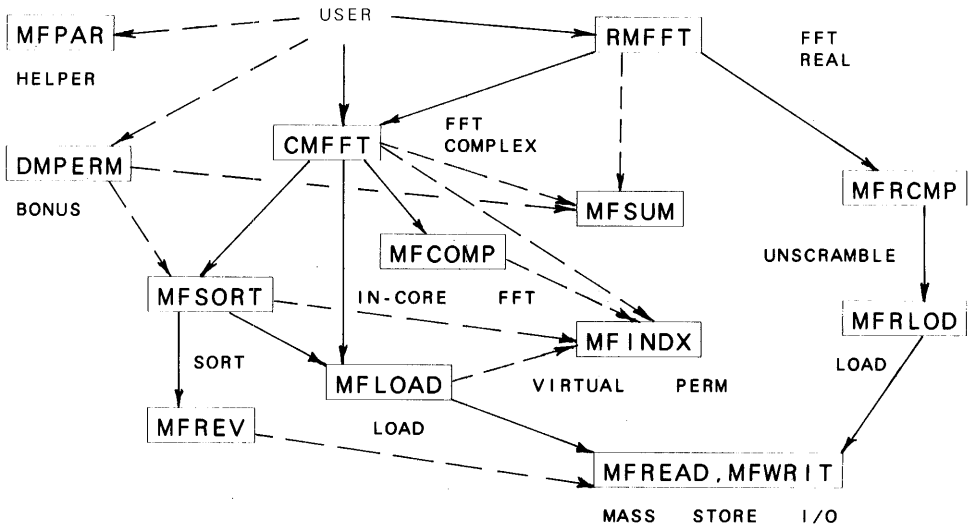


Fig. 1. Internal subroutine calling network. (Solid and dashed lines distinguished roughly between "main path" and "reference" calls.)

array before unscrambling by MFRCMP. If necessary, routines CMFFT and below may be overlaid with routines MFRCMP and MFRLOD. CMFFT controls the sequence of mass store accessing for in-core computation and sorting of the complex FFT. Routines MFCOMP and MFRCMP carry out the FFT computation and unscrambling of data in core store. MFREV, controlled by routine MFSORT, carries out a bit-reversed permutation on elements in core store or on whole blocks in mass store. MFLOAD and MFRLOD load and unload core store for the in-core routines, calling on MFREAD and MFWRIT (Appendix F). MFINDX defines a virtual permutation of the mass store array during the complex FFT operation. MFSUM sums elements in the dimensioning list MEXA( ), sorting them if necessary. Routine MFPAR is discussed in Appendix B and routine DMPERM in Appendix D.

The main problem in writing a mass store FFT program is to devise an efficient means of accessing widely spaced data elements for use in the FFT computation kernel and for sorting from bit-reversed order [2]. Singleton [9] describes an algorithm for accessing a mass store array two blocks at a time to carry out each computing pass of an FFT. He organizes mass store accesses in such a way that each I/O and computing pass results in a one-place cyclic shift of array element index bits. This allows the FFT kernel program to access the same elements in core store on each pass, which not only simplifies the program but also brings "widely separated" elements in the original array "within reach" of the in-core computation. Bit-reversed sorting is done in a similar manner. For an array of 2**M elements, the basic method requires (2M − 1) I/O passes.

With efficient, random access mass store it is possible to use an indexing algorithm to read widely scattered blocks, equivalent to a virtual permutation of the array. After an in-core operation the blocks are written back in place, the blocks remaining physically unpermuted by the operation. In [8] an algorithm is described which uses this method to compute the FFT. The indexing subroutine MFINDX computes the required sequence of block indices using an algorithm which is a generalization of the Block Indexing Algorithm of [8, p. 303]. This is equivalent to a cyclic shift (a number of places) of a set of the block index bits.

Complex Data. Unlike Singleton's algorithm, an attempt is made to load as many blocks together as possible in core store. In this way it is possible to include a number of FFT computing passes while the blocks are in core store, reducing the number of I/O passes by a factor ICEX − IBEX (see the conclusion section). Routine CMFFT uses a slightly modified version of the algorithm of [8, p. 307] for this purpose, doing M − C instead of M − B initial computing passes with

virtual permutations, followed by final C pass computations with direct access, instead of final B pass computations. I/O efficiency is unchanged, but the algorithm is neater (sizes being 2**B elements per block, 2**C in core store, 2**M total, all complex).

Routine MFCOMP computes the FFT in core store. But, unlike other FFT routines, the number of computing passes NPAS, and the effective initial pass number IPAS, are not fixed but are passed by the calling routine CMFFT. Weighting factors are computed using SIN, COS functions on mass store block boundaries but recursively within block boundaries, while array element accessing is ordered to minimize these computations. Multidimensional transforms are achieved by restarting the weighting factor sequence over passes corresponding to the exponent of each dimension, requiring no change to the order of access of elements.

Finally, routine MFSORT is called to sort the array from bit-reversed order. The algorithm is a slightly modified version of the sorting algorithm of [8, pp. 305–307], to allow the reversal of a more general set of index bits. Its structure is very similar to CMFFT, having a first stage using the virtual cyclic shift permutation for block access, with pairs of in-core bit-reversed permutations (MFREV) instead of FFT computations. (The pairs allow unsymmetric cyclic bit shift permutations to be done in place, in core). Only one in-core permutation is required on the first I/O pass. These are followed by a final bit-reversed permutation of whole blocks, if necessary.

In the special case when IDIR = 0, MFSORT is called a number of times to individually reverse bits corresponding to each dimension exponent. This allows the complex FFT to be done without dimension reversal (but note that dimension reversal is required by RMFFT). The dimension shifting subroutine DMPERM, of Appendix D, also uses this feature. In both these cases, I/O is not fully optimized and specially written programs could reduce the number of I/O passes by combining some of the separate, in-core permutations.

Bit-reversed index pairs are generated by a very efficient recursion algorithm (routine MFREV). The algorithm maintains a hierarchy of "reversed" integers of increasing number of bits, up to 2 less than the number being reversed. Incrementing a reversed integer then requires the alternate simple addition of a constant or replacement by the next lower incremented reversed integer in the hierarchy, recursively. For example, with a 3-bit reversal, the reversed set is (0, 4, 2, 6, 1, 5, 3, 7) where the next value is obtained either by adding 4 to the last or by replacing it by one of the values (0, 2, 1, 3) of a 2-bit reversed set.

This method of reversed series generation is in itself fast, as recursion depths are small on the average. But, in addition, only quarter-length series are generated (−2 bits) and the full-length series is derived by scaling by 2 and adding offsets. This is equivalent to reversing an integer a(integer − 2 bits)b to b(reversed integer − 2 bits)a where a and b are the outer bits. There are four possible combinations for ab, but only those reversals greater numerically than before reversal are required (to prevent nullifying double swaps), leaving in general the three offsets $1\cdots0$, $0\cdots0$, and $1\cdots1$. In particular, only the first offset is required if the internal (reversed integer − 2 bits) is smaller than or the same as before reversal. Thus, only valid swap index pairs are generated, saving the unnecessary reversed integer generation of some other methods.

Real Data. For transforms in which the initial or final data is real, a half-length complex transform of packed real data must be unscrambled (or vice versa) by routine MFRCMP. The method relies on complex-conjugate symmetry in the transform of real data [4]. Calculation of indices of multidimensional symmetry is more difficult than in the one-dimensional case. To do this a recursion algorithm is used, which is most easily described by the following Fortran program (for two dimensions, N1-by-N2 elements):

```
L2 = N2/2 + 1
DO 1 J2 = 1, L2
```

```
      K2 = N2 + 2 - J2
      IF(J2.EQ.1)K2 = 1
      L1 = N1
      IF(J2.EQ.K2)L1 = N1/2 + 1
      DO 1 J1 = 1, L1
      K1 = N1 + 2 - J1
      IF(J1.EQ.1)K1 = 1
C     Now have A(K1, K2) and A(J1, J2) as a pair with conjugate symmetry
    1 CONTINUE
```

Note is taken of the special cases which exist when the J index is 1 (Fortran) and when the J and K indices are equal. The general case follows by repetition of the code between the two DO statements, replacing J2 by Jj, etc. In routine MFRCMP arrays JAYA(4), etc., are used for this purpose. To increase the number of dimensions allowed in RMFFT (Appendix A), increase the array sizes. To help visualize the result, the index relationships (Fortran index −1) for an 8-by-4 array are as follows (r for real IF(index(K1, K2).EQ.index(J1, J2)), where index(J1, J2) = J1 + (J2 −1)*N1, for example, c for complex conjugate):

r00	01	02	03	r04	c03	c02	c01
10	11	12	13	14	15	16	17
r20	21	22	23	r24	c23	c22	c21
c10	c17	c16	c15	c14	c13	c12	c11

The array above represents the transformation and transposition of a 4-by-8 real array (imaginary part zero). However, we may also consider it to represent the scrambled transform of an array of initially packed real data, occupying alternately real and imaginary elements, that is, initially 8-by-8 real values. In this case the result is only half-length, element pairs such as 15 and c15 above containing information which can be unscrambled to give a new element 15 and an element 33 in place of c15. In addition, new elements c15 and c33 are derived. Thus the array is expanded to an 8-by-8 full transform of the real data:

r00	01	02	03	r04	c03	c02	c01
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37
r40	41	42	43	r44	c43	c42	c41
c30	c37	c36	c35	c34	c33	c32	c31
c20	c27	c26	c25	c24	c23	c22	c21
c10	c17	c16	c15	c14	c13	c12	c11

Note that the same indexing algorithm applies in both examples (with different N2) but during unscrambling and expansion only the half-length symmetry is used (N2 = 4); corresponding expanded rows in the new array are obtained by a direct offset (4 in this case). Thus, by half-length symmetry, initial rows 1 and 3 are accessed together for unscrambling; these are replaced by unscrambled rows 1 and 3 of the full array while new rows 5 and 7 are also computed and added to the array. It is this ability to add data beyond the existing data which makes dimension reversal essential for in-place computation in the real mass store FFT routine RMFFT.

The complete array is "fully redundant," nearly half the elements being complex conjugates of the other half. If the last three rows of the example are left out, the result is "partially redundant," since row 0 and row 4 still have some internal redundancy. To eliminate all redundancy, rows 0 and 4 can be merged (c43, c42, c41 replacing c03, c02, c01), and pairs of real elements combined as single complex elements (r40 as the imaginary part with r00 real, r44 with r04).

The exact algorithm by which an array with IPAK = 0 or −1 (see Appendix A) can be restored to full redundancy is given by the half-length symmetry program above, putting

```
A(J1, J2 + N2) = CONJG(A(K1, K2))
A(K1, K2 + N2) = CONJG(A(J1, J2)),   IF(J2.NE.1)
```

with special provision IF(J2.EQ.1) when IPAK = −1:

```
A(K1, K2 + N2) = A(K1, K2)
A(J1, J2 + N2) = CONJG(A(K1,K2))
A(K1, K2) = CONJG(A(J1, J2))
```

and IF(index(K1, K2).EQ.index(J1, J2)) when IPAK = −1, values are real:

```
A(J1, J2 + N2) = CMPLX(AIMAG(A(J1, J2)), 0.)
A(J1, J2) = CMPLX(REAL(A(J1, J2)), 0.)
```

the general case follows with Jj and Nj instead of J2 and N2 and as before with repetition of the code between DO statements.

Conclusion

The number of I/O passes through the mass store array depends on the array size (2**M), the core store working space size (2**ICEX), and the I/O block or record size (2**IBEX). In general, the larger the working space and the smaller the block size the better, but block size should not be made too small because of other overheads. The FFT computation requires

$$\mathcal{I}((M - IBEX + 1)/(ICEX - IBEX))$$

passes involving I/O (where $\mathcal{I}(x)$ is the smallest integer greater than or equal to x). Postcomputation sorting requires a similar number of passes (without sorting overlap), although DeLotto and Dotti [5] mention

$$\mathcal{I}((M - ICEX + 1)/(ICEX - IBEX))$$

passes. The reason this is not achieved is that in-place sorting requires M − IBEX + 1 virtual permutations to leave a physically unpermuted array. A change in the algorithm to a smaller number M − ICEX + 1 virtual permutations requires an extra block-sorting pass of mass store, nullifying the advantage.

Figure 2 gives the maximum range of block size exponent IBEX necessary to keep the number of I/O passes N of the mass store array low, given ICEX and M. Solid lines bound the optimum N = 4. Dashed lines are the boundaries for the next best N = 6 passes, and it should not be difficult to operate with N = 4 or 6 in most cases. When calling the half-length transform by subroutine RMFFT, increase diagram M and N by 1 to obtain the working M and N.

The hatched line cutting across the figure gives a bound below which all mass store passes access all blocks. In the area above the line,

$$\mathcal{I}((IBEX - 1)/(ICEX - IBEX))$$

sorting passes are required with an additional pass of mass store involving a block reshuffle, in which only some of the blocks are accessed. That is, the optimum can approach N = 3 passes. Given an I/O system which allows alteration of the index key of a block, the block shuffle can be replaced by an index shuffle, giving N − 1 passes in the upper area. Note also that the diagram is only approximate, as integral rounding effects may increase or decrease the number of passes by one at some points.

Run time depends on two main factors, the computation time of the in-core FFT and the I/O time. A total elapsed time can be written approximately

$$T = TC*M*2**M + TM*N*2**M$$

where TC and TM are unit computation and unit mass store average access and transfer times corresponding to each complex element. As an example, consider two very different computer systems, a PDP 11/40 system and a CYBER 76 installation. Then we may expect TC = 0.5 ms (PDP 11) or 2 μs (CYBER), and TM = 1 ms (PDP 11, IBEX = 7) or 0.5 ms (CYBER, IBEX = 9) for routine CMFFT. Calling subroutine RMFFT with packed real data roughly halves the time, so that a 256-by-256 real array can be transformed in about 5 min (PDP 11) or 1 s (CYBER) CPU time and 8 min (PDP 11) or 1 min (CYBER) elapsed time.

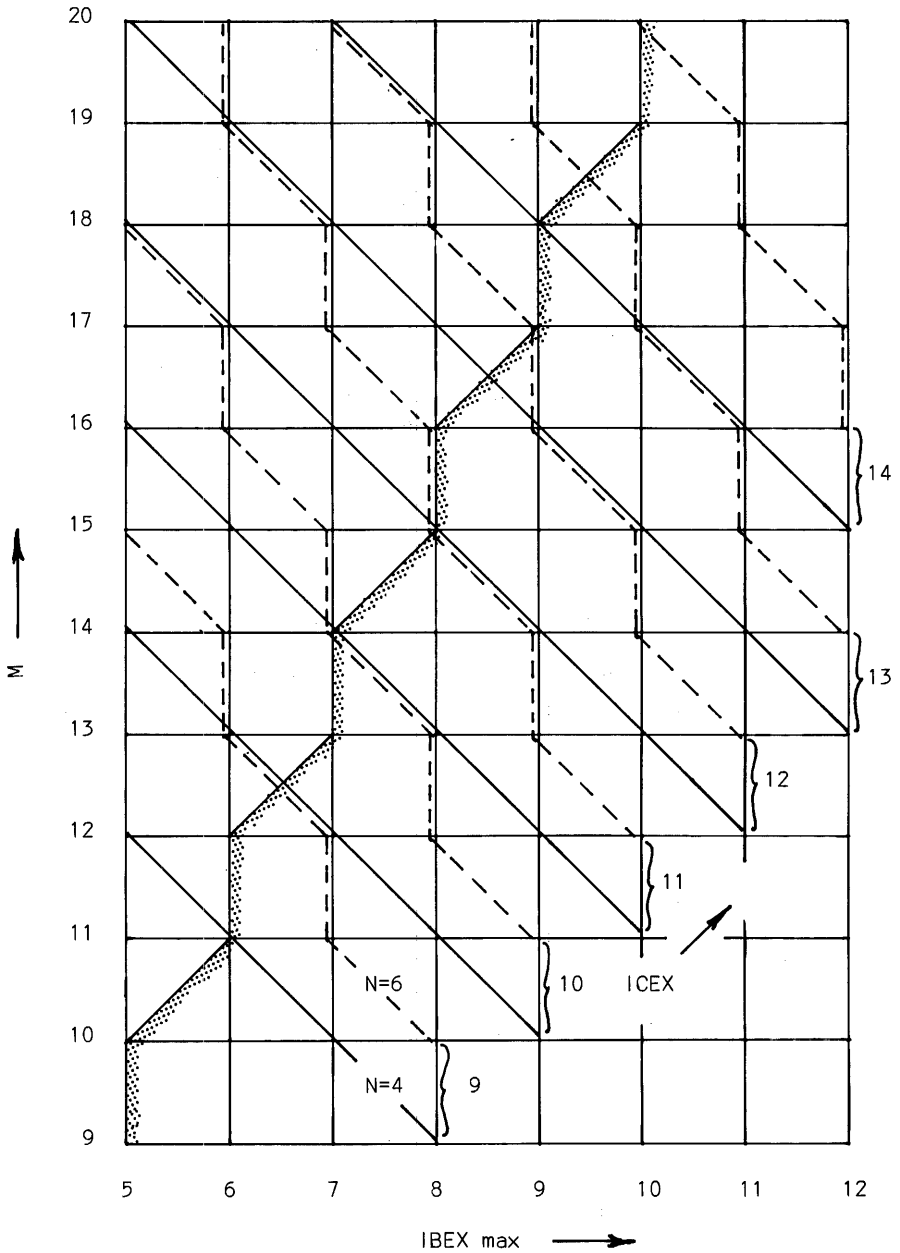


Fig. 2. Mass store I/O passes N related to binary exponents IBEX, ICEX, and M for routine CMFFT. Increase diagram values of M and N by 1 to obtain working M and N for routine RMFFT.

But in machines such as the CYBER, where computing speed is very great compared to I/O latency, the routines are used to great advantage for transforming arrays in LCM (extended memory) in place of mass store, using high-speed, block-copy operations between this and main memory (see Appendixes E and F). Whether an array is multidimensional or not makes little difference to efficiency.

Both TC and TM are block size dependent, there being fixed overheads per block. Both therefore can be written

$$TC = TCO + TCA/2^{**}(IBEX - 1), \qquad TM = TMO + TMA/2^{**}(IBEX - 1)$$

where TCA is the FFT computation overhead per block and TMA is the average mass store access time per block. TCO and TMO are the limits for large block size. TCA and TMA set a useful lower limit for block size. In the examples above TM is mainly the result of TMA, or system block access time. Increasing block size reduces TM but with a probable increase in N (depending on M and ICEX).

TC is mainly due to TCO, TCA becoming important typically when IBEX < 5. To determine the most efficient parameter combination, begin by choosing ICEX as large as possible; use Figure 2 as a guide in choosing IBEX but follow this up by timing comparisons over a range of IBEX.

The program was written to test the sorting algorithm [8] in practice. At the same time a useful and efficient Fourier program has been obtained, operating with a very general set of parameters. Some parameter combinations can lead to short cuts, not detected or implemented here (see the section, Algorithm Details). Efficiency in a particular case can no doubt be improved by some recoding at the expense of a loss in generality or increase in code complexity.

Hopefully this program will stimulate designers not only of fast transforms but also of general purpose systems. Sorting by index bit manipulation should be considered an important concept for new machine architecture. Index bit reversal is simple to achieve in hardware and, in its generalized form [8], is a symmetric permutation (allowing in-place operation) which forms the basis of a number of useful permutations, not exclusively associated with the FFT.

Appendix A. Program Usage

The complex FFT routine is called by

```
CALL CMFFT (MEXA, NDIM, ISGN, IDIR, SCAL, BUFA, IBEX, ICEX)
```

while the faster, real-to-complex (or vice versa) FFT routine is called by

```
CALL RMFFT (MEXA, NDIM, ISGN, IDIR, SCAL, BUFA, IBEX, ICEX, IPAK)
```

where the subroutine arguments have the following meanings:

MEXA Integer array of size NDIM (defined below). MEXA consists of a list of dimension size binary exponents, defining the dimensioning of the mass store array. For example, a one-dimensional array has a size of 2**MEXA(1) elements. A two-dimensional array has 2**MEXA(2) sets of 2**MEXA(1) adjacent elements each (N1 and N2 of definition equations (1) or (2) are 2**MEXA(1) and 2**MEXA(2) in initial order).

Notes: (1) The FFT routines actively modify the MEXA list, if necessary, leaving it in the order corresponding to the final array dimensioning. In general, its order is reversed by the FFT, except in the special case with routine CMFFT discussed under parameter IDIR below. The MEXA list should therefore be linked to a unique mass store array so that it always indicates the current dimensioning of that array.

(2) The MEXA list of exponents always refers to the dimensioning of a full array, as required by routine CMFFT or by routine RMFFT with IPAK = +1 (defined below). The complex result of routine RMFFT operating on real data is truncated when IPAK = 0 or −1, so that the MEXA list elements in these cases may refer to a virtual array length.

(3) The first element of the list always gives the size 2**MEXA(1) of the set or sets of adjacent elements in the first dimension of the mass store array. If data are of type complex, this is the number of complex elements in each set while if data are of type real (e.g., before calling routine RMFFT in direction real to complex), this is the number of real elements.

(4) Mass store array dimensioning is independent of I/O block transfer size and core working space size.

NDIM Number of dimensions in mass store array (size of MEXA list). Range $1 \leq NDIM \leq 4$ (RMFFT), to increase (see the section, Algorithm Details), or $1 \leq NDIM \leq M$ (CMFFT).

(M) Not a call parameter, but defined here for convenience. M = sum to NDIM of MEXA list, giving total mass store array size = 2**M elements

(number of complex elements for routine CMFFT, real or complex elements for routine RMFFT; but see under IPAK packing parameter below).

ISGN The sign of ISGN is the sign of the complex exponent of the transform definition (e.g., eq. (1) or (2)). A positive or negative sign results in the inverse transform of the other; but see also below.

IDIR Transform direction (RMFFT), reversal (CMFFT) parameter. Routine RMFFT converts packed real data to complex data (or vice versa) during transformation, so that IDIR is needed to determine the direction, independently of ISGN; thus

IDIR = −1, real to complex (RMFFT), dimension reversal;
IDIR = +1, complex-to-real (RMFFT), dimension reversal.

(Note that it is usually most convenient to use the same variable for both ISGN and IDIR so that a single negation results in transform inversion.)

Routine CMFFT does not need a direction parameter, other than ISGN, and IDIR is used in this case to set dimension reversal (transposition in two dimensions) or not, as required, thus

IDIR not zero, dimension reversal (more efficient I/O);
IDIR = 0, do not use (RMFFT), suppress reversal (CMFFT).

SCAL Arbitrary type real scale factor of eq. (1) or (2). If SCAL = 1.0 computation is fastest as no scaling occurs.

BUFA Array in core to be used as workspace by FFT routines. Note that internal FFT subroutines assume the following: (1) BUFA is either type real or complex, to suit local needs. (2) BUFA is given the trivial dimension BUFA(1) internally, since its actual size is known only as the exponent ICEX (defined below). (3) Type complex data are assumed to be stored in the sequence real/imaginary/real/imaginary/... in core store and mass store. Some of these points may upset some Fortran compilers.

IBEX Mass store I/O block transfer size binary exponent.
Block or record size = 2**IBEX real elements.

Limits are $2 \le IBEX \le ICEX - 2$ (RMFFT),
or $1 \le IBEX \le ICEX - 1$ (CMFFT).

ICEX Core store working space size binary exponent.
Dimension of BUFA = 2**ICEX real elements.

Limits are $IBEX + 2 \le ICEX \le M$ (RMFFT),
or $IBEX + 1 \le ICEX \le M + 1$ (CMFFT).

IPAK Packing parameter (routine RMFFT only). Determines the degree of redundancy (discussed further in the section, Algorithm Details) desired in the complex result after a real-to-complex transform; thus:

IPAK = +1 gives a fully redundant complex result. The final mass store array is exactly twice the physical length of the initial real array, having the same number, 2**M, of complex elements as initial real elements. This is the same result that is obtained when calling CMFFT with the initial data occupying the real part of a complex array, zero imaginary.

IPAK = 0 gives a partly redundant complex result, slightly longer than the initial real array. In this case there are 2**(M − 1) + 2**(M − MEXA(NDIM)) complex elements in the final array, the MEXA list in the order after the transform (increased to an integral number of mass store blocks, if necessary). This is probably the most useful packing.

IPAK = −1 gives a result containing no redundancy and having exactly the same physical length as the initial 2**M real elements, or 2**(M − 1) complex elements. This is achieved by squeezing together those parts of the array cussed in the section, Algorithm Details. No information is lost and, for one or two dimensions, only a little sorting is needed to access unsqueezed information. For example, in one dimension, the real value at the Nyquist frequency becomes the imaginary part of the zero frequency element, which also must be real.

Notes: (1) For IPAK = +1 or 0 the mass store array file must be extendable. IPAK = −1 has the advantage that the mass store array file remains a fixed length, but has the disadvantage that some user effort is needed to access complex data correctly.

(2) In the complex-to-real direction, IPAK = +1 or 0 are equivalent, since only part of a fully redundant complex array is accessed by RMFFT in this direction. But IPAK = −1 must be used in both directions to correctly handle the squeezed complex array.

Appendix B. Helper Routine

To help the user set up the arguments of Appendix A and to determine the mass store file and block sizes, a helper routine MFPAR is included. Use of the routine is not essential, but is recommended. The routine is called by

```
IERR = MFPAR (IRMF, ICOMP)
```

with IRMF = −1 if mass store data are currently packed real or +1 if data are currently complex when using routine RMFFT. When using routine CMFFT, IRMF = 0. If ICOMP = 0 the argument exponents MEXA( ), IBEX, and ICEX are defined by the user while if ICOMP is not zero the exponents are to be computed by MFPAR.

Three COMMON blocks are used to transmit other data; thus

```
COMMON/MFARG/MEXA(4), NDIM, ISGN, IDIR, SCAL, IBEX, ICEX, IPAK
COMMON/MFVAL/DIMA (4), TDM1, RDM1, FBLK, TBLK, RBLK, RCOR, SIZE
COMMON/MFINT/ NDMA(4), NTD 1, NRD1, NFBK, NTBK, NRBK, NRCR, NSZE
```

MFARG holds a four-element MEXA( ) list (for up to four dimensions) followed by the other mass store FFT call arguments (except BUFA). NDIM, IBEX, ICEX, and IPAK must be preset (unless ICOMP not zero), while ISGN, IDIR, and SCAL are ignored here.

COMMON blocks MFVAL and MFINT return data computed by routine MFPAR. These include the four-element arrays DIMA( ) and NDMA( ) corresponding element by element to MEXA( ) but containing the actual dimension sizes 2**MEXA( ). Similarly, RBLK, NRBK, RCOR, and NRCR hold the sizes 2**IBEX and 2**ICEX. Note that MFINT variables are one-to-one integer conversions of MFVAL variables, which are of type real. A local variable FIXMAX determines whether a conversion is allowed (set to 32767. by a data statement in MFPAR, but which may be altered to suit). Any values not converted are set to −1 in MFINT and the function returns MFPAR = −1 to indicate this. IBEX and ICEX are forced to be within the limits defined in Appendix A, and MFPAR = +1 if IBEX is forced too small. Otherwise the function returns MFPAR = 0 normally.

If the helper subroutine argument ICOMP is not zero, the computation is reversed and MEXA( ) exponents are computed from given DIMA( ) real sizes, IBEX and ICEX computed from RBLK and RCOR real sizes. Sizes are adjusted to be integral powers of 2, adjusted up or down to the closest power on a log scale. NDIM and IPAK must still be given in MFARG.

The most useful values computed by MFPAR are NTD1, NRD1, NFBK, NTBK, and NRBK (in MFINT). NRBK is 2**IBEX and is the number of reals in the "record" used for mass store access by the FFT. NTBK is the current total number of records of size NRBK or the current file length. NFBK, on the other hand, is the maximum number of records of size NRBK to be expected, including any mass store array expansion by routine RMFFT (see IPAK = 0 or 1 in Appendix A). NFBK is thus useful for defining the maximum file length to the operating system.

NRD1 is the number of reals in an equivalent record of the current first dimension length ("first" defined by current MEXA(1)). NTD1 is the current number of such equivalent records and these two values are useful for logically accessing a multidimensional mass store array by the user (though not so useful if NDIM = 1). Note that to do this the I/O routines MFREAD and MFWRIT must be able to handle correctly records different in length from NRBK; otherwise NTBK and NRBK should be used. This will be system dependent (see Appendix F).

NSZE, SIZE is the effective total size of the mass store array (2**M) and is useful, for example, in computing the scale factor SCAL.

Note that MFPAR should be called just prior to user file access, and with the correct value for IRMF, to ensure the computed variables reflect the current state of the mass store array.

Appendix C. Example

Suppose we wish to transform a two-dimensional real array having 512 rows (2**9) of 256 adjacent real elements each (2**8), or a total of 128K real elements (2**17), where K = 1024. Suppose we decide to allow a core store working space of 8K real elements (2**13) and to access the mass store array in blocks of 128 real elements (2**7) each, these being quite independent of array dimensioning. Then, using the helper routine MFPAR (Appendix B) to compute file parameters, the mass store file is defined and opened for random access, loaded with data, and subroutine RMFFT is called with

```
      COMMON/MFARG/MEXA(4), NDIM, ISGN, IDIR, SCAL, IBEX, ICEX, IPAK
      COMMON/MFVAL/DIMA(4), TDM1, RDM1, FBLK, TBLK, RBLK, RCOR, SIZE
      COMMON/MFINT/NDMA(4), NTD1, NRD1, NFBK, NTBK, NRBK, NRCR, NSZE
C     COMMON BLOCKS USED BY HELPER ROUTN MFPAR (NOT ESSENTIAL)
C
      REAL BUFA(8192)
      COMPLEX CBUFA(4096)
      EQUIVALENCE (BUFA(1), CBUFA(1))
C     WORKING ARRAY IN CORE (EQUIVALENCED FOR USER ACCESS)
C
      MEXA(1) = 8
      MEXA(2) = 9
      NDIM = 2
      ISGN = -1
      IDIR = -1
      SCAL = 1.0
      IBEX = 7
      ICEX = 13
      IPAK = 1
C     MASS STORE FFT ARGUMENTS INITIALIZED
C
      IRMF = - 1
      ICOMP = 0
      IERR = MFPAR (IRMF, ICOMP)
C     HELPER ROUTINE, DATA REAL, RMFFT, EXPONENTS DEFINED,
C     IF(IERR.EQ.0) OK
C     COMPUTES FILE PARAMETERS IN COMMON AREAS (NOT ESSENTIAL)
```

(open mass store file here)

```
C     HERE CAN OPEN MASS STORE FILE, NFBK MAX 'RECDS' OF NRBK REALS
C
```

```
      DO 2 JB = 1, NTD1                    (or NTBK)
      DO 1 I = 1, NRD1                     (or NRBK)
1     BUFA(I) = (Enter a real variable here)
2     CALL MFWRIT (BUFA, NRD1, JB)         (or (BUFA, NRBK, JB))
C   LOAD MASS STORE FILE WITH PACKED REAL DATA
C   (IF MFREAD/MFWRIT REQUIRE FIXED RECD LENGTH, USE NTBK, NRBK)
C
```

CALL RMFFT (MEXA, NDIM, ISGN, IDIR, SCAL, BUFA, IBEX, ICEX, IPAK)

resulting in the transform of eq. (2), with SCAL = 1.0, N1 = 256, N2 = 512, and a negative complex exponent. Because the initial data are packed real and because the parameter IPAK = 1 is chosen, the mass store array will be extended to 128K complex elements, or twice its initial physical size. This array is a fully redundant transform of the original, having 256 rows (2**8) of 512 adjacent complex elements (2**9), the dimensions being reversed or transposed. To indicate transposition to the MEXA list will be reversed (MEXA(1) = 9, MEXA(2) = 8 after the transform).

To save unnecessary array extension, the last parameter IPAK can be made 0 or −1. If IPAK = 0 the result is a partially redundant transform, being the first 129 rows (2**MEXA(2)/2 + 1) of the transform, of 512 adjacent complex elements each. The first and last rows each have an internal conjugate symmetry, but other redundancy is deleted.

If IPAK = −1 the first and last rows above are "squeezed" together, the second half of the last row becoming the second half of the first row. In addition, the first and middle real elements of the last row become the imaginary parts of the first and middle elements of the first row, resulting in an array of 128 rows of 512 complex elements, exactly the same physical length as the initial array, so that no file extension is necessary (see the section, Algorithm Details).

The complex result is accessed, then an inverse transform is called by

```
      IRMF = 1
      IERR = MFPAR (IRMF, ICOMP)
C   HELPER ROUTINE AGAIN, DATA COMPLEX, ROUTINE RMFFT,
C   IF(IERR.EQ.0) OK
C
      NCNT = NRD1/2                        (or NRBK/2)
C   NCNT IS THE NUMBER OF COMPLEX ELEMENTS IN RECORD
      DO 4 JB = 1, NTD1                    (or NTBK)
      CALL MFREAD (BUFA, NRD1, JB)         (or (BUFA, NRBK, JB))
      DO 3 I = 1, NCNT

3     (access each complex element CBUF(I) here)

4   CALL MFWRIT (BUFA, NRD1, JB)           (or (BUFA, NRBK, JB))

C   COMPLEX RESULT READ BY USER (WRITING NEW VALUES IF DESIRED)
C   (IF MFREAD/MFWRIT REQUIRE FIXED RECD LENGTH, USE NTBK, NRBK)
C
      ISGN = −ISGN
      IDIR = −IDIR
      SCAL = 1.0/SIZE
      CALL RMFFT (MEXA, NDIM, ISGN, IDIR, SCAL, BUFA, IBEX, ICEX, IPAK)
```

The scale factor SCAL, using SIZE = 2.**M computed by MFPAR, is chosen here to normalize the result to the same scale as the original data. The result is a real array having 512 rows of 256 adjacent real elements each, the MEXA list being restored to its initial order.

Appendix D. Bonus Routine

The sorting algorithm used in the mass storage FFT may also be used for changing the order of dimensioning of a multidimensional mass store array. This is similar to, but more general than, Eklundh's method [7]. As an example, and in addition to the FFT routines, a dimension-shifting routine DMPERM is included for completeness, called by

```
CALL DMPERM (MEXA, NDIM, NSHFT, IREX, BUFA, IBEX, ICEX)
```

where the arguments MEXA, NDIM, BUFA, IBEX, ICEX have the same meaning as in the FFT calls.

The other parameters are

NSHFT	Dimension shift count.	
	NSHFT = 0	No shift or change occurs.
	NSHFT = 1, 2 etc.	First-to-next dimension, circular NSHFT place shift, modulo (NDIM).
	NSHFT = −1	Dimension order reversed.
IREX	"Element size" binary exponent (size = 2**IREX reals); thus IREX = 0 for real array, IREX = 1 for complex array.	

Most of the comments concerning the FFT routine parameters apply also to this routine. The mass store array is either real or complex (or "elements" of multiple reals), with an MEXA list defining the current dimensioning (binary exponents). The MEXA list is actively modified by the routine if necessary. (Note that to sort arrays of elements smaller in size than type real, such as the type byte elements of some systems, it is necessary to alter type statements for BUFA and TEMP variables in all subroutines called.)

The routine is quite trivial in design, consisting of only 20 statements. It operates by calling the generalized mass store bit-reversed sorting routine MFSORT, used internally by the FFT routines, and therefore has a similar I/O efficiency to the FFT routines (see the conclusion section).

Appendix E. Test Programs

A universal test program is provided which sets up, for a given M, exhaustive permutations and combinations of the different parameters IBEX, ICEX, and NDIM from one to three dimensions. Mass store is simulated, through dummy routines MFREAD and MFWRIT, by a core store array so that the program is independent of system I/O.

The program stores a pseudorandom number sequence in the simulated mass store, transforms this by the mass store FFT routines, and compares the result with a discrete Fourier transform computed naively by routine NAIVE. An inverse mass store FFT is called and the result compared with the initial data. Maximum differences are noted and various levels of data and difference printout are available through parameter IPRINT. The test is considered successful if maximum differences are explainable in terms of machine roundoff errors.

Because the tests are exhaustive and because of the naive DFT computation the program is quite slow when the overall array size exponent M is other than very small. For example, in a CYBER 76 computer with M = 5 the program takes 0.3 s while with M = 10 the program takes about 800 s. Less exhaustive testing can be done by replacing program DO-loop variables by fixed variables (see program comments).

Example test programs are also included for accessing mass store in two specific computer systems. These are a CYBER 76 (true mass store and LCM "static" mass store) and a PDP 11. Mass store I/O routines for other systems should be easily devised based on these examples and the comments of Appendix F.

Appendix F. System-Dependent I/O Routines

The system-dependent random access transfers between mass store and core store are handled by two subroutines called internally by the FFT routines; thus

```
CALL MFREAD (BUFA, NB, JB)      read block into core store
CALL MFWRIT (BUFA, NB, JB)      write block from core store
```

where BUFA is the core store address for the start of the transfer, NB is the number of real elements to be transferred, and JB is the desired block or record number. The range of JB is as follows:

for CMFFT or RMFFT if IPAK = +1,

$1 \le JB \le 2**(M - IBEX + 1),$

for RMFFT if IPAK = 0,

$1 \le JB \le 2**(M - IBEX) + 2**(M - MEXA(NDIM) - IBEX + 1),$ for $MEXA(NDIM) + IBEX < M + 1,$

or

$1 \le JB \le 2**(M - IBEX) + 1,$ for $MEXA(NDIM) + IBEX \ge M + 1,$

(the MEXA list in the order after the transform)

for RMFFT if IPAK = −1,

$1 \le JB \le 2**(M - IBEX).$

In addition, the I/O subroutines must know what file to access and any starting block offset, which may be given through COMMON variables, for example. The routines expect the mass store file to be defined and opened. The user program may, of course, call MFREAD and MFWRIT itself to load mass store with data or read the transform result.

For example, in PDP 11 Fortran a random access occurs with

```
      SUBROUTINE MFREAD (BUFA, NB, JB)
C   READ BLOCK, INDEX JB, FROM MASS STORE TO BUFA, NB REAL VALUES
      REAL BUFA(NB)
      COMMON/FFTCOM/LUN
      READ (LUN'JB)(BUFA(I), I = 1, NB)   or   READ (LUN'JB) BUFA
      RETURN
      END
```

and similarly for subroutine MFWRIT.

As discussed in Appendixes B and C it may be useful to call MFREAD/MFWRIT with a record length NB different from 2**IBEX. This can give a user more logical access to a multidimensional array, in records of first dimension length, for example. Some systems allow a redefinition of the file structure, in which case the problem is solved. Others allow a "word-addressable" file structure in which an I/O transfer may start at any word in the file. In this case, MFREAD/MFWRIT can include the following:

$INDEX = (JB - 1)*NB + 1$

(transfer between file real element indices (INDEX) and (INDEX + NB − 1). The dummy I/O routines of the universal test program of Appendix E and the CYBER Extended Core/LCM and PDP 11 Macro routines include this feature.

Subroutines MFREAD and MFWRIT allow the user considerable scope for modifying the operation of the FFT routines. In the universal test driver program (Appendix E) the subroutines simply copy data from one core store array to another. In some systems external core store can be accessed efficiently in blocks, which allows this to be used as a fast, static mass store. Using different files on the first I/O pass for MFREAD and MFWRIT allows data to be copied automatically from a source file to a working and result file. In this case care must be taken to allow only a single read of each block of the source file, any subsequent read, even on the "first" I/O pass, being from the working file (important during complex-to-real transformation by routine RMFFT).

REFERENCES

1. BRENNER, N.M. FOR2D. Program 360D-13.4.007, SHARE Program Library Agency, 1969.
2. COCHRAN, W.T., ET AL., What is the fast Fourier transform? *IEEE Trans. Audio Electroacoust. AU-15,* 2 (June 1967), 45–55.
3. COOLEY, J.W., AND TUKEY, J.W. An algorithm for the machine calculation of complex Fourier series. *Math Comput. 19* (April 1965), 297–301.
4. COOLEY, J.W., ET AL., The fast Fourier transform algorithm: Programming considerations in the calculation of sine, cosine, and Laplace transforms. *J. Sound Vib. 12* (July 1970), 315–337.
5. DELOTTO, I., AND DOTTI, D. A new procedure for optimum mass storage use in FT algorithms. *Alta Freq. 42,* 8 (Aug. 1973), 379–384.

6. DeLotto, I., and Dotti, D. Two-dimensional transforms by minicomputer without matrix transposing. *Comp. Graphics Image Processing 4* (Oct. 1975), 271-278.
7. Eklundh, J.O. A fast computer method for matrix transposing. *IEEE Trans. Comptr. C-21,* 7 (July 1972), 801-803.
8. Fraser, D. Array permutation by index-digit permutation. *J. ACM 23,* 2 (April 1976), 298-309.
9. Singleton, R.C. A method for computing the fast Fourier transform with auxiliary memory and limited high-speed storage. *IEEE Trans. Audio Electroacoust. AU-15,* 2 (June 1967), 91-98.

ALGORITHM

```
C
C     THIS SET OF PROGRAM AND SUBROUTINE UNITS IS TO SUPPORT
C
C       'AN OPTIMIZED MASS STORAGE FFT',  BY DONALD FRASER
C
C           REVISION DATE: JULY 1978 (MINOR REV JUNE 79).
C
C     THE SET INCLUDES A UNIVERSAL TEST DRIVER PROGRAM, WHICH SIMULATES
C  MASS STORE THROUGH A FORTRAN ARRAY, SAMPLE PROGRAMS AND I/O
C  SUBROUTINES FOR CONTROL DATA 6ØØØ AND CYBER COMPUTERS AND FOR
C  DEC PDP 11 MINICOMPUTERS.
C
C     I/O SUBROUTINES FOR OTHER SYSTEMS SHOULD BE EASILY CONSTRUCTED
C  FROM THE EXAMPLES AND WITH REFERENCE TO THE FORMAL PAPER.
C  BUT CARE SHOULD BE TAKEN WITH SOME FUSSY COMPILERS SINCE FFT
C  SUBROUTINES ASSIGN EITHER TYPE REAL OR TYPE COMPLEX TO THE SAME
C  ARRAY AS IT IS PASSED AS A FORMAL PARAMETER.  NOTE THAT COMPLEX
C  DATA IS ASSUMED TO BE STORED  REAL/IMAG/REAL/IMAG... IN BOTH
C  MASS STORE AND CORE STORE.
C
C     THE PROGRAM UNITS APPEAR IN THE FOLLOWING ORDER:
C
C         FIRST, THE FFT SUBROUTINE SET:
C  1  RMFFT        OPTIMIZED MASS STORAGE FFT (REAL DATA OR REAL RESULT)
C  2  CMFFT        CALLED BY 1, OR MASS STORAGE FFT (ALL COMPLEX DATA)
C  3  MFCOMP       IN-CORE FFT
C  4  MFSORT       MASS STORE SORTING
C  5  MFREV        IN-CORE SORTING OR WHOLE BLOCK SORTING
C  6  MFLOAD       LOADING/UNLOADING CORE STORE
C  7  MFINDX       BLOCK INDEXING ALGORITHM (VIRTUAL PERMUTATION)
C  8  MFSUM        MEXA() EXPONENT SUMMATIONS
C  9  MFRCMP       REAL-COMPLEX UNSCRAMBLING/SCRAMBLING
C 1Ø  MFRLOD       LOADING/UNLOADING CORE STORE FOR MFRCMP
C 11  MFPAR        HELPER ROUTINE (NOT ESSENTIAL, BUT RECOMMENDED)
C 12  DMPERM       MASS STORE DIMENSION SHIFTING (BONUS SUBROUTINE)
C
C          THEN, TEST PROGRAMS AND SAMPLE I/O SUBROUTINES
C
C 13  UNIVERSAL TEST PROGRAM (SIMULATED MASS STORE, NEEDS 14 TO 17 ALSO)
C 14  RANMF        RANDOM NUMBER GENERATOR
C 15  NAIVE        DISCRETE FOURIER TRANSFORM COMPUTED NAIVELY
C 16  MFREAD       DUMMY I/O ROUTINE USING SIMULATED MASS STORE
C 17  MFWRIT          AS ABOVE
C
C 18  CYBER MASS STORE SAMPLE PROGRAM
C 19  MFREAD/MFWRIT  CYBER MASS STORE I/O ROUTINE
C
C 2Ø  CYBER EXTENDED CORE/LCM SAMPLE PROGRAM
C 21  MFREAD/MFWRIT  CYBER EXTENDED CORE/LCM I/O ROUTINES
C
C 22  PDP 11 MASS STORE SAMPLE PROGRAM
C 23  MFREAD       PDP 11 STANDARD FORTRAN MASS STORE I/O ROUTINES
C 24  MFWRIT          AS ABOVE
C
C 25  PDP 11 FAST MACRO I/O SAMPLE PROGRAM (NEEDS MACRO OPEN SUBRTN)
C 26  MFREAD/MFWRIT  PDP 11 FAST MACRO I/O ROUTINE FOR RSX11M/RT11.
C
      END


      SUBROUTINE RMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX, IPAK)
C
C REAL-TO-COMPLEX FFT (OR VICE-VERSA) OF MULTI-DIMENSD MASS STORE ARRAY
C  (FRASER, ACM TOMS - 1978/79, AND J.ACM, V.23,N 2, APRIL 76, PP. 298-3Ø9)
```

```
C  MASS STORE ARRAY IS EITHER REAL OR COMPLEX DATA (SEE NOTE BELOW)
C
C  NOTE WELL THAT TYPE COMPLEX DATA MUST EXIST AS ALTERNATING
C  REAL/IMAG/REAL/IMAG... ELEMENTS, BOTH IN MASS STORE AND IN FORTRAN
C  WORKING ARRAY BUFA;  IN THIS FFT, DIFFERENT SUBROUTINES WILL SET
C  DIFFERENT TYPE (REAL OR COMPLEX) FOR ARRAY BUFA.
C
C MEXA(J) LIST OF DIMENSION SIZE EXPONS (BASE 2), ADJACENT VARIABLES FIRST
C NDIM IS NUMBER OF EXPONENTS IN LIST AND THUS THE NUMBER OF DIMENSIONS
C  SUM TO NDIM OF MEXA(J) = M, WHERE 2**M IS EFFECT SIZE OF MASS STORE ARRAY
C     THUS, 2**M PACKED REAL VALUES,
C     OR,   2**M COMPLEX VALUES, IF COMPLEX RESULT WITH IPAK=1 (SEE BELOW)
C RMFFT ALWAYS REVERSES DIMENSION ORDER AND MEXA LIST (TRANSPOSED, IF 2 D)
C
C ISGN GIVES SIGN OF COMPLEX EXPONENT OF TRANSFORM (+ OR -), AND
C IDIR DETERMINES DIRECTION OF TRANSFORM, THUS:
C     IDIR=-1,  REAL-TO-COMPLEX
C     IDIR=+1,  COMPLEX-TO-REAL
C SCAL IS REAL MULTIPLIER OF RESULT (EG. SET SCAL=1. FWD, 1./2**M INV)
C
C BUFA IS CORE STORE WORKING ARRAY BASE ADDRESS (SEE NOTE ABOVE)
C IBEX, ICEX ARE BLOCK AND CORE SIZE EXPONENTS, THUS
C  2**IBEX IS NUMBER OF REAL ELEMENTS IN MASS STORE BLOCK
C  2**ICEX IS NUMBER OF REAL ELEMENTS IN CORE STORE BUFA
C
C IPAK IS ARRAY PACKING DETERMINATOR, THUS:
C  IPAK=+1 EXPANDS COMPLEX ARRAY TO FULL REDUNDANCY (SAME AS SUBRTN CMFFT)
C  IPAK=Ø  COMPUTES COMPLEX ARRAY OF JUST OVER HALF SIZE (COMMON METHOD)
C  IPAK=-1 HOLDS COMPLEX ARRAY AT EXACTLY HALF SIZE (2**(M-1) CMPLX ELMTS)
C
C MASS STORE ARRAY MUST BE OPEN FOR ACCESS BY USER SUBRTNS MFREAD/MFWRIT:
C  EG. SUBRTN MFREAD(BUFA,NB,JB) AND MFWRIT(BUFA,NB,JB) TRANSFER ONE
C  BLOCK, INDEX JB, BETWEEN MASS STORE AND CORE STORE BUFA, NB REALS
C  (1.LE.JB.LE.2**(M-IBEX) IF REAL, OR 2**(M-IBEX+1) IF CMPLX AND IPAK=1)
C
      COMPLEX BUFA(1)
      INTEGER B,MEXA(1)
C
      MH=MFSUM(MEXA,NDIM,99)-1
      B=IBEX-1
      IF(IDIR.GT.Ø)GO TO 1Ø
C
C BELOW, REAL-TO-COMPLEX TRANSFORM
      MEXA(1)=MEXA(1)-1
      CALL CMFFT  (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX)
      CALL MFRCMP (MEXA,NDIM,ISGN,IDIR,IPAK, BUFA,B,MH)
      MEXA(NDIM)=MEXA(NDIM)+1
      RETURN
C
C BELOW, COMPLEX-TO-REAL TRANSFORM
1Ø    MEXA(NDIM)=MEXA(NDIM)-1
      CALL MFRCMP (MEXA,NDIM,ISGN,IDIR,IPAK, BUFA,B,MH)
      CALL CMFFT  (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX)
      MEXA(1)=MEXA(1)+1
      RETURN
C
      END


      SUBROUTINE CMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX)
C
C COMPLEX FFT OF MULTI-DIMENSD MASS STORE ARRAY, CALLED BY USER OR RMFFT
C  FOR COMMENTS, SEE SUBRTN RMFFT; ARGUMNTS HAVE SAME MEANING EXCEPT FOR
C  IDIR=+1 OR -1, ARRAY ALWAYS COMPLEX, DIMENSION ORDER REVERSED
C  WHILE IDIR=Ø, DIMENSION ORDER (AND MEXA LIST) ARE NOT REVERSED
C
      COMPLEX BUFA(1)
      INTEGER B,C,MEXA(1)
      DATA LSET/1/,LREAD/1/,LWRIT/2/
C
      M=MFSUM(MEXA,NDIM,99)
      MREAL=M+1
      B=IBEX-1
      C=ICEX-1
      NC=2**C
      IPAS=Ø
```

```
      MPAS=M-C
      NPAS=C-B
C MOST EFFICIENT USE OF CORE STORE - TRIES TO DO C-B PASSES PER LOAD
      IDUM=MFINDX(LSET,B,M,M,NPAS)
C DUMMY CALL TO MFINDX TO SPECIFY VIRTUAL (B.'S'.M)**(C-B) PERMUTATION
C
C FIRST, PIECE-MEAL ATTACK ON FFT COMPUTATION FOLLOWS
1Ø    IF(MPAS.LE.Ø) GO TO 4Ø
      IF(MPAS.LT.NPAS)NPAS=MPAS
2Ø    CALL MFLOAD(LREAD,BUFA,IBEX,ICEX,IFLG)
C LOAD CORE WORKING SPACE WITH 2**(C-B) BLOCKS ACCORDING TO MFINDX
      IF(IFLG.LT.Ø) GO TO 3Ø
      CALL MFCOMP (MEXA,NDIM,ISGN, BUFA,B,C,M, NPAS,IPAS)
C DO MODIFIED IN-CORE FFT, REQUIRING NPAS PASSES STARTING WITH IPAS
      CALL MFLOAD(LWRIT,BUFA,IBEX,ICEX,IFLG)
C UNLOAD CORE AREA, WRITING BLOCKS BACK IN-PLACE TO MASS STORE
      GO TO 2Ø
3Ø    IPAS=IPAS+NPAS
      MPAS=MPAS-NPAS
      GO TO 1Ø
C END OF FIRST PART
C
C SPECIFY BLOCKS TO BE READ IN NEXT PART IN TRUE ORDER (NO PERM)
4Ø    IDUM=MFINDX(LSET,B,M,M,Ø)
C FINAL, CONCLUDING ATTACK ON FFT COMPUTATION FOLLOWS
5Ø    CALL MFLOAD(LREAD,BUFA,IBEX,ICEX,IFLG)
      IF(IFLG.LT.Ø)GO TO 8Ø
      CALL MFCOMP (MEXA,NDIM,ISGN, BUFA,C,C,M, C,IPAS)
C DO FINAL, C-PASS IN-CORE FFT OF EACH CORE-LOAD
      IF(SCAL.EQ.1.)GO TO 7Ø
      DO 6Ø J=1,NC
6Ø    BUFA(J)=BUFA(J)*SCAL
7Ø    CALL MFLOAD(LWRIT,BUFA,IBEX,ICEX,IFLG)
      GO TO 5Ø
C
C BELOW, SORT ARRAY (FULL BIT-REVERSAL AND DIMEN REVERSAL IF IDIR.NE.Ø)
8Ø    IF(IDIR.EQ.Ø)GO TO 9Ø
      CALL MFSORT(BUFA,IBEX,ICEX,1,MREAL,MREAL)
      M=MFSUM(MEXA,NDIM,-1)
C DO FULL BIT-REVERSAL OF M BITS (AND REVERSE MEXA LIST)
      RETURN
C
C BELOW, REVERSE BITS OF EACH DIMEN SEPARATELY (NO DIMEN REVERSAL)
9Ø    IH=1
      DO 1ØØ J=1,NDIM
      IG=IH
      IH=IH+MEXA(J)
1ØØ   CALL MFSORT(BUFA,IBEX,ICEX,IG,IH,MREAL)
      RETURN
C
      END


      SUBROUTINE MFCOMP (MEXA,NDIM,ISGN, BUFA,B,C,M, NPAS,IPAS)
C
C MODIFIED, IN-CORE FFT OF 2**C ELEMENTS, NPAS PASSES STARTING WITH IPAS
C  MEXA, NDIM, ISGN ,BUFA AND M HAVE SAME MEANING AS IN RMFFT COMMENTS
C  B,C EQUIVALENT TO IBEX,ICEX EXCEPT HERE REFER TO NUM COMPLEX ELMTS
C  (2**C CMPLX ELMTS IN CORE STORE BUFA, IN BLOCKS OF 2**B CMPLX ELMTS)
C
C FFT W PHASE FACTOR COMPUTED RECURSIVELY EXCEPT ON BLOCK BOUNDS
C  MULTIDIMEN FFT ACHIEVED BY REPEATING W SEQUENCES
C
      INTEGER B,C,SPAN,STEP,MEXA(1)
      COMPLEX TEMP,W,D,BUFA(1)
      DATA LINDX/Ø/,LREST/4/
C
      DATA PI/3.141592653589793/
      PIMOD=PI*2.Ø**(1-M)
      IF(ISGN.LT.Ø)PIMOD=-PIMOD
      NC=2**C
C
C BELOW, NPAS COMPUTATION PASSES WHILE DATA IN-CORE
      DO 5Ø JPAS=1,NPAS
      KPAS=IPAS+JPAS-1
C KPAS IS GLOBAL COMPUTING PASS NUMBER (JPAS-1 IS LOCAL)
```

```
      KDIFF=M-MFSUM(MEXA,NDIM,KPAS)
      KPEFF=KPAS+KDIFF
C KPEFF IS EFFECTIVE GLOBAL PASS FOR MULTIDIMEN. FFT W GENERATION
      D=CEXP(CMPLX(Ø.,PIMOD*2.Ø**KPEFF))
C D IS USED FOR RECURSIVE MODIFICATION OF W PHASE FACTOR
C
      ITEM=C-JPAS
      SPAN=2**ITEM
      STEP=2*SPAN
C SPAN SEPARATES VALUES IN FFT KERNEL, STEP TO NEXT PAIR, SAME W
C
      IF(B.LT.ITEM)ITEM=B
      NB=2**ITEM
      IF(ITEM.GT.KDIFF)ITEM=KDIFF
      NRPT=2**ITEM
C NRPT COUNTS REPETITION OF W FOR MULTIDIMEN. FFT
C
      IMOD=2**(M-B-KPAS-1)
      IF(IMOD.LE.1)GO TO 2Ø
      IDUM=MFINDX(LREST,Ø,Ø,Ø,Ø)
      MEXP=KPEFF
      ITEM=KDIFF-B
      IF(ITEM.GT.Ø)GO TO 1Ø
      MEXP=B+KPAS
      ITEM=Ø
1Ø    NCLR=2**ITEM
      PIMOD2=PIMOD*2.Ø**MEXP
C IMOD AND NCLR ARE USED TO COMPUTE W WITHOUT EXCEEDING SMALL INTEGER
C
C BELOW, START OF ONE PASS THROUGH CORE, NOTING BLOCK BOUNDARIES
2Ø    DO 5Ø I1=1,SPAN,NB
      W=(1.,Ø.)
      IF(IMOD.LE.1)GO TO 3Ø
      INDWM=MOD(MFINDX(LINDX,Ø,Ø,Ø,Ø)-1,IMOD)/NCLR
      ANDWM=INDWM
      W=CEXP(CMPLX(Ø.,PIMOD2*ANDWM))
C NEW W COMPUTED DIRECTLY AT BEGINNING OF NEW BLOCK AREA
C
C BELOW, COMPUTATIONS WITHIN EACH BLOCK OF 2**B CMPLX ELMTS
3Ø    DO 5Ø I2=1,NB,NRPT
C
C BELOW, REPETITION OF SAME W DUE TO MULTIDIMEN FFT
      DO 4Ø I3=1,NRPT
      I4=I1+I2+I3-2
C
C BELOW, STEPPING THOUGH INDICES HAVING SAME W IN ONE DIMEN FFT
      DO 4Ø J=I4,NC,STEP
      K=J+SPAN
      TEMP=(BUFA(J)-BUFA(K))*W
      BUFA(J)=BUFA(J)+BUFA(K)
4Ø    BUFA(K)=TEMP
C FFT 2-POINT KERNEL ARITHMETIC (ALGORITHM BIT-REVERSAL FOLLOWS COMPUT)
C
      W=W*D
C RECURSIVE MODIFICATION OF W WITHIN BLOCK BOUNDARIES
5Ø    CONTINUE
      RETURN
C
      END


      SUBROUTINE MFSORT(BUFA,IBEX,ICEX,IG,IH,M)
C
C BIT-REVERSED PERMUTATION (IG.'R'.IH) OF MASS STORE REAL ARRAY
C  REVERSES IH-IG BITS IN INDEX (M-1,...,IH-1,...,IG,...,Ø)
C  NOTE THAT THIS IS MORE GENERAL THAN THE FULL M-BIT REVERSAL
C  OF REFERENCE (FRASER, J.ACM, V.23, N.2, APR. 76, P. 3Ø6),
C  BUT ALGORITHM IS LOGICALLY THE SAME, WITH ALTERED BIT LIMITS.
C BUFA,IBEX,ICEX AND M HAVE SAME MEANING AS IN COMMENTS IN RMFFT
C  (BLOCKS 2**IBEX, CORE BUFA 2**ICEX, TOTAL 2**M, ALL REAL)
C
      REAL BUFA(1)
      DATA LSET/1/,LPERM/2/,LREAD/1/,LWRIT/2/
C
      IDUM=MFINDX(LSET,IBEX,M,M,Ø)
C DUMMY CALL TO INITIALISE MFINDX (INITIALLY UNPERMUTED ARRAY)
```

```
      IF(IH-IG.LE.1)RETURN
      IF(IG.GE.IBEX)GO TO 5Ø
      IF(IH.LE.ICEX)GO TO 6Ø
C CHECK FOR SPECIAL CASES, REQUIRING SIMPLER TREATMENT
C
C BELOW, MIXED PERMUTATION OF BOTH ELEMENTS AND BLOCKS
      IPAS=Ø
      NPAS=ICEX-IBEX
C MOST EFFIENT USE OF CORE STORE - TRIES TO DO ICEX-IBEX PASSES PER LOAD
      MPAS=IH-IBEX
      IF(IBEX-IG.LT.MPAS)MPAS=IBEX-IG
C
C BELOW, FIRST VIRTUAL 'S' PERMUTATIONS
1Ø    IF(MPAS.LE.Ø)GO TO 4Ø
      IF(MPAS.LT.NPAS) NPAS=MPAS
      IGCOR=IG+IPAS
      IF((IGCOR.GT.IBEX-1).AND.(IGCOR.GT.IBEX+NPAS-1))GO TO 3Ø
      IF((IPAS.EQ.Ø).AND.(IGCOR.GT.IBEX+NPAS-1))GO TO 3Ø
C BYPASS UNNECESSARY CORE LOAD IF TRIVIAL CASES
      IDUM=MFINDX(LPERM,IBEX,IH,M,NPAS)
C DUMMY CALL TO MFINDX TO SPECIFY VIRTUAL (IBEX.S.IH)**NPAS PERM
C
C BELOW, LOAD CORE ACCORDING TO VIRTUAL PERMUTATION AND PERM ELMTS
2Ø    CALL MFLOAD(LREAD,BUFA,IBEX,ICEX,IFLG)
      IF(IFLG.LT.Ø)GO TO 3Ø
      IF(IPAS.NE.Ø) CALL MFREV(BUFA,IGCOR,IBEX,ICEX,-1)
      CALL MFREV(BUFA,IGCOR,IBEX+NPAS,ICEX,-1)
C CARRY OUT IN-CORE, SYMMETRIC R PERMS. (ONE ONLY ON FIRST PASS)
      CALL MFLOAD(LWRIT,BUFA,IBEX,ICEX,IFLG)
C UNLOAD CORE AREA, WRITING BLOCKS BACK IN-PLACE TO MASS STORE
      GO TO 2Ø
C
3Ø    IPAS=IPAS+NPAS
      MPAS=MPAS-NPAS
      GO TO 1Ø
C END OF FIRST PART
C
C BELOW, FINAL 'R' PERM. OF BLOCKS IN MASS STORE, IF (IH-2*IBEX+IG).GT.1
4Ø    CALL MFREV(BUFA,Ø,IH-2*IBEX+IG,M-IBEX,IBEX)
      RETURN
C
C BELOW, PERMUTATION OF BLOCKS ONLY REQUIRED
5Ø    CALL MFREV(BUFA,IG-IBEX,IH-IBEX,M-IBEX,IBEX)
      RETURN
C
C BELOW, PERMUTATION OF ELEMENTS IN CORE ONLY REQUIRED
6Ø    CALL MFLOAD(LREAD,BUFA,IBEX,ICEX,IFLG)
      IF(IFLG.LT.Ø)RETURN
      CALL MFREV(BUFA,IG,IH,ICEX,-1)
      CALL MFLOAD(LWRIT,BUFA,IBEX,ICEX,IFLG)
      GO TO 6Ø
C
      END


      SUBROUTINE MFREV(BUFA,IG,IH,M,IBEX)
C
C BIT-REVERSED PERMUTATION OF RANDOMLY ADDRESSABLE ELEMENTS
C  REVERSES IH-IG BITS IN INDEX (M-1,...,IH-1,...,IG,...,Ø)
C  (GENERAL PERM IG.'R'.IH, FRASER, J.ACM, V.23, N.2, APR 1976, P. 3ØØ)
C IF IBEX.LT.Ø, SORTS 2**M REAL ELMTS IN CORE BUFA,
C IF IBEX.GE.Ø, SORTS BLOCKS IN MASS STORE
C (2**IBEX REAL ELMTS PER BLOCK AND 2**M BLOCKS IN SECOND CASE)
C
C  THE ALGORITHM MAINTAINS A SET OF 'REVERSED' INTEGERS IN ARRAY IRA()
C  OF INCREASING NUMBER OF BITS, UP TO 2 LESS THAN (IH-IG) BITS.
C  INCREMENTING A REVERSED INTEGER THEN REQUIRES THE ALTERNATE
C  ADDITION OF NRA() TO IRA(), OR REPLACEMENT BY THE NEXT LOWER
C  INCREMENTED REVERSED INTEGER IN THE HIERARCHY, RECURSIVELY.
C  THIS IN ITSELF IS FAST, AS RECURSION DEPTHS ARE ON AVERAGE SMALL.
C  BUT, IN ADDITION, ONLY QUARTER LENGTH SERIES ARE GENERATED (-2 BITS)
C  AND THE FULL LENGTH DERIVED BY SCALING BY 2 AND ADDING OFFSETS.
C  IN THIS FINAL STAGE, ONLY VALID SWAP PAIRS ARE GENERATED (1 OR 3 EACH)
C
C  WITHIN THE INNER LOOPS, GROUPS OF 2**IG ELMTS ARE MOVED TOGETHER
C  WHILE THIS IS REPEATED OVER 2**(M-IH) PARTS OF THE ARRAY,
```

```
C  CORRESPONDING TO THE UNPERMUTED BITS M TO IH AND IG-1 TO Ø.
C
      REAL BUFA(1),TEMP
      INTEGER IRA(16),NRA(16),IFOFA(3),IROFA(3)
C
      IHG=IH-IG-3
      IF(IHG.LE.-2)RETURN
C NO PERMUTATION REQUIRED
C
      NB=2**IBEX
      NB1=NB+1
      NG=2**IG
      NGDB=NG*2
      NH=2**IH
      NHHF=NH/2
C NG IS MOVEMENT GROUP SIZE, NH IS PERMUTATION REPLICATION SIZE
      NM=2**M
      NPARS=NM-NH+1
      NREV=NH/4
C
      DO 1Ø J=1,IHG
      IRA(J)=Ø
      NREV=NREV/2
1Ø    NRA(J)=NREV
C REVERSED INTEGER RECURSION SETS INITIALISED
C
      NREV=NH/4
      IFOFA(1)=NG-1
      IROFA(1)=NH/2-1
      IFOFA(2)=-1
      IROFA(2)=-1
      IFOFA(3)=NH/2+NG-1
      IROFA(3)=NH/2+NG-1
C THREE PAIRS OF OFFSETS TO CONVERT QUARTER TO FULL LENGTH SERIES
C
      IFOR=Ø
      IREV=Ø
C
C BELOW, GENERATE INDEX PAIRS AND SWAP (IREV IS 'TOP' OF IRA() SET)
2Ø    NOF=3
      IF(IFOR.GE.IREV)NOF=1
C SELECTS ONCE-ONLY SWAP PAIRS (EITHER 1 OR 3 PAIRS)
      DO 4Ø JOF=1,NOF
      IFOF=IFOFA(JOF)
      IROF=IROFA(JOF)
      DO 4Ø I1=1,NG
C REPETITION OVER GROUP OR SUPER ELEMENT OF NG ACTUAL ELEMENTS
      IN2F=IFOR+IFOF+I1
      IN2R=IREV+IROF+I1
      DO 4Ø I2=1,NPARS,NH
C REPETITION OF SAME PERMUTATION OVER ARRAY PARTS
      IN3F=IN2F+I2
      IN3R=IN2R+I2
      IF(IBEX.GE.Ø)GO TO 3Ø
C
C BELOW, IN-CORE ELEMENT SORTING
      TEMP=BUFA(IN3R)
      BUFA(IN3R)=BUFA(IN3F)
      BUFA(IN3F)=TEMP
      GO TO 4Ø
C
C BELOW, SORTING WHOLE BLOCKS IN MASS STORE
3Ø    CALL MFREAD(BUFA,NB,IN3F)
      CALL MFREAD(BUFA(NB1),NB,IN3R)
      CALL MFWRIT(BUFA,NB,IN3R)
      CALL MFWRIT(BUFA(NB1),NB,IN3F)
C
4Ø    CONTINUE
C END OF INNER, REPETITION LOOPS
C
      IFOR=IFOR+NGDB
C INCREMENT FORWARD QUARTER-LENGTH INTEGER (ALREADY SCALED BY NG*2)
      IF(IFOR.GE.NHHF)RETURN
C RETURN FORM SUBROUTINE
C
      IF(IREV.GE.NREV)GO TO 5Ø
```

```
C TEST FOR ALTERNATE METHODS OF REVERSE-INCREMENTING (SIMPLE BELOW)
C NOTE THAT REVERSE QUARTER-LENGTH INTEGER IS ALREADY SCALED BY NG*2
C
      IREV=IREV+NREV
      GO TO 2Ø
C
C ALTERNATE RECURSIVE ALTERATION TO QUARTER-LENGTH REVERSED SERIES
5Ø    DO 6Ø J=1,IHG
      IF(IRA(J).LT.NRA(J))GO TO 7Ø
6Ø    CONTINUE
C
C BELOW, SIMPLE INCREMENT OF REVERSE INTEGER, LOWER IN HIERARCHY
7Ø    IRA(J)=IRA(J)+NRA(J)
      IREV=IRA(J)
8Ø    IF(J.EQ.1)GO TO 2Ø
      J=J-1
      IRA(J)=IREV
      GO TO 8Ø
C
      END


      SUBROUTINE MFLOAD(LOAD,BUFA,IBEX,ICEX,IFLG)
C
C LOADS, UNLOADS CORE STORE ARRAY BUFA, 2**ICEX REALS, 2**IBEX PER BLOCK
C  RETURNS IFLG=+1 NORMALLY, IFLG=-1 WHEN FINISHED ONE PASS OF MASS STORE
C BLOCKS INDEXED ACCORDING TO VIRTUAL PERMUTATION FUNCTION MFINDX
C LOAD=1 (LREAD) READS BLOCKS FROM MASS STORE INTO CORE STORE BUFA
C LOAD=2 (LWRIT) WRITES BLOCKS BACK IN-PLACE TO MASS STORE
C
      REAL BUFA(1)
      DATA LINDX/Ø/,LHOLD/3/,LREST/4/
C
      NB=2**IBEX
      NCB=2**(ICEX-IBEX)
      IF(LOAD.EQ.2)GO TO 3Ø
      IFLG=+1
      IDUM=MFINDX(LHOLD,Ø,Ø,Ø,Ø)
C HOLDS CURRENT MFINDX VALUE FOR ENTRY 2 AND SUBRTN MFCOMP
      DO 1Ø J=1,NCB
      K=(J-1)*NB
      JB=MFINDX(LINDX,Ø,Ø,Ø,Ø)
      IF(JB.LT.Ø)GO TO 2Ø
      CALL MFREAD(BUFA(K+1),NB,JB)
C READS BLOCK WITH NEXT VIRTUAL MFINDX INDEX
1Ø    CONTINUE
      RETURN
2Ø    IFLG=-1
      RETURN
C
3Ø    IDUM=MFINDX(LREST,Ø,Ø,Ø,Ø)
C RESETS MFINDX TO START OF IN-PLACE BLOCK
      DO 4Ø J=1,NCB
      K=(J-1)*NB
      JB=MFINDX(LINDX,Ø,Ø,Ø,Ø)
      CALL MFWRIT(BUFA(K+1),NB,JB)
C WRITES BLOCK WITH NEXT VIRTUAL MFINDX INDEX (REPEAT MFREAD SEQUENCE)
4Ø    CONTINUE
      RETURN
C
      END


      FUNCTION MFINDX(LSPEC,B,H,M,N)
C
C VIRTUAL 'S' PERMUTATION (FRASER, J.ACM, V.23, N.2, APR. 76, P.3Ø3)
C  CYCLIC SHIFTS H-B BITS IN INDEX (M-1,...,H-1,...,B,...,Ø)
C COMPUTES NEXT INDEX FOR SEQUENTIAL CORE LOAD, PERM (B.'S'.H)**N
C BLOCK SIZE EXPON B, MASS STORE EXPON M (Ø.LE.B.LE.H.LE.M)
C N IS EFFECTIVE NUMBER OF LEFT SHIFTS PER I/O PASS (-N RIGHT SHIFTS)
C
C NOTE VARIABLE NAMES AS FOLLOWS:
C     IPERM IS 'P' OF ALGORITHM
C     NPERM=N (ARGUMENT) IS 'N' OF ALGORITHM
C     ISTEP IS 'Q**P' OF ALGORITHM
```

```
C     JAY AND KAY ARE 'J' AND 'K' OF ALGORITHM
C NOTE UPPER BOUND H INSTEAD OF M, REQUIRING 2**(M-H) REPEATS
C
C LSPEC=Ø (LINDX) RETURNS MFINDX FOR INDEX (B,H,M,N DUMMIES HERE)
C LSPEC=1 (LSET) SETS IPERM=Ø (UNPERMED), ENTERS B,H,M,N PARAMS
C LSPEC=2 (LPERM) CHANGES THE B,H,M,N PARAMETERS
C LSPEC=3 (LHOLD) HOLDS CURRENT INDEXING STATE (B,H,M,N DUMMIES HERE)
C LSPEC=4 (LREST) RESTORES STATE TO LAST LHOLD (B,H,M,N DUMMIES HERE)
C
      INTEGER B,H
C
      IF(LSPEC.EQ.1)GO TO 1ØØ
      IF(LSPEC.EQ.2)GO TO 2ØØ
      IF(LSPEC.EQ.3)GO TO 3ØØ
      IF(LSPEC.EQ.4)GO TO 4ØØ
C
1Ø    IF(ISTEP.NE.Ø)GO TO 2Ø
C BELOW, PRECEDES FIRST MFINDX OF A PASS
      IF(NPERM.GT.Ø)IPERM=MOD(IPERM-NPERM,IHB)
      IF(IPERM.LT.Ø)IPERM=IHB+IPERM
      ISTEP=2**IPERM
C
C 2Ø BELOW, NORMAL GENERATION OF NEXT MFINDX
2Ø    MFINDX=JAY+JOFF
      IF(MFINDX.GT.NMB)GO TO 4Ø
      KAY=JAY+ISTEP
      JAY=MOD(KAY,NHB)
      IF(KAY.GE.NHB)JAY=JAY+1
      NRPT=NRPT-1
      IF(NRPT.GT.Ø)RETURN
C
C NRPT,JOFF REQUIRED TO REPEAT SEQUENCE ON 2**(M-H) PARTS OF ARRAY
      JOFF=JOFF+NHB
3Ø    NRPT=NHB
      JAY=Ø
      RETURN
C
C 4Ø BELOW, END OF ONE PASS, PARS RESET, IPERM ALTERED IF INVERSE
4Ø    IF(NPERM.LT.Ø)IPERM=MOD(IPERM-NPERM,IHB)
      MFINDX=-1
5Ø    JOFF=1
      ISTEP=Ø
      GO TO 3Ø
C
C LSPEC=1 (LSET) SETS IPERM=Ø (UNPERMED), ENTERS B,H,M,N PARAMS
1ØØ   IPERM=Ø
C
C LSPEC=2 (LPERM) CHANGES THE B,H,M,N PARAMETERS (DUMMIES ELSEWHERE)
2ØØ   IHB=H-B
      NPERM=N
      NHB=2**IHB
      NMB=2**(M-B)
      MFINDX=IPERM
      GO TO 5Ø
C
C LSPEC=3 (LHOLD) HOLDS CURRENT MFINDX INDEXING PARAMETERS
3ØØ   JAYH=JAY
      JOFH=JOFF
      NRPTH=NRPT
31Ø   MFINDX=IPERM
      RETURN
C
C LSPEC=4 (LREST) RESTORES PARAMETERS TO INDEX MFINDX AT LAST LHOLD
4ØØ   JAY=JAYH
      JOFF=JOFH
      NRPT=NRPTH
      GO TO 31Ø
C
      END


      FUNCTION MFSUM(MEXA,NDIM,MLIM)
C
C SCANS MEXA LIST IN REVERSE ORDER, RETURNING (MFSUM.JUST GT.MLIM)
C  (IF MLIM LARGE ENOUGH, RETURNS M TOTAL FOR NDIM VALUES)
```

```
C  (IF MLIM NEGATIVE, RETURNS M TOTAL, REVERSES ORDER OF MEXA LIST)
C
      INTEGER MEXA(1)
C
      MFSUM=Ø
      IF(NDIM.LE.Ø)RETURN
C
      DO 1Ø J=1,NDIM
      I=NDIM+1-J
      MFSUM=MFSUM+MEXA(I)
      IF((MLIM.GE.Ø).AND.(MLIM.LT.MFSUM))RETURN
1Ø    CONTINUE
      IF(MLIM.GE.Ø)RETURN
C
C BELOW, REVERSE ORDER OF MEXA LIST
      NDIMH=NDIM/2
      DO 2Ø J=1,NDIMH
      K=NDIM+1-J
      MTEM=MEXA(J)
      MEXA(J)=MEXA(K)
2Ø    MEXA(K)=MTEM
      RETURN
C
      END


      SUBROUTINE MFRCMP (MEXA,NDIM,ISGN,IDIR,IPAK, BUFA,B,M)
C
C UNSCRAMBLES REAL-TO-COMPLEX FFT OR VICE-VERSA, CALLED BY SUBRTN RMFFT
C  MOST ARGUMENTS HAVE SAME MEANING AS IN RMFFT COMMENTS
C  BUT 2**B COMPLEX ELMTS IN MASS STORE BLOCK,
C  USES (2**B)*4 CMPLX IN BUFA, 'LOWER', 'UPPER' PLUS EXPANSION AREAS
C  TOTAL MASS STORE ARRAY SIZE OF 2**M COMPLEX ELMTS.
C
      COMPLEX ATEM,BTEM,TEMP,W,D,BUFA(1)
      INTEGER B,JAYA(4),KAYA(4),JWKA(4),KWKA(4),MEXA(1)
C JAYA,KAYA,JWKA,KWKA ALLOW UP TO 4 DIMENSIONS - INCREASE IF REQUIRED
      DATA PI/3.141592653589793/
      DATA LOWER/1/,LUPPR/2/,LCLR/-1/
C
      DO 1Ø IDIM=1,NDIM
      JAYA(IDIM)=Ø
      KAYA(IDIM)=Ø
1Ø    CONTINUE
C MULTIDIMEN. CONJUGATE-SYMMETRIC INDICES ZEROED
C
      IEXPND=1
      NB=2**B
      NBDB=NB*2
      JBOF=2**(M-B)
      MAX=M-B-MEXA(NDIM)
      IF(IDIR*IPAK.LT.Ø)MAX=M-B
      JBMAX=2**MAX
      IF(MAX.LT.Ø)JBMAX=1
      IF(IPAK.LT.Ø)JBMAX=Ø
C JBMAX IS MAXIMUM BLOCK INDEX REQUIRED (DEPENDS ON IPAK)
C
      IWFG=Ø
      W=(1.,Ø.)
      D=CEXP(CMPLX(Ø.,PI*2.Ø**(-MEXA(NDIM))))
      IF(ISGN.LT.Ø)D=CONJG(D)
C W IS COMPLEX PHASE FACTOR, D IS RECURSIVE MODIFIER OF W
C
2Ø    JAY=Ø
      KAY=Ø
      IDIM=NDIM
3Ø    IF(IDIM.EQ.1)GO TO 4Ø
      NUMD=2**MEXA(IDIM)
      JWKA(IDIM)=JAY
      KWKA(IDIM)=KAY
      JAY=JAY*NUMD+JAYA(IDIM)
      KAY=KAY*NUMD+KAYA(IDIM)
      IDIM=IDIM-1
      GO TO 3Ø
C CONJUGATE-SYMMETRIC BASE INDICES COMPUTED FROM MULTIDIMEN. SET
C
```

```
4Ø    MEX1=MEXA(1)
C 2**MEX1 IS NUMBER OF VALUES ADJACENT IN FIRST DIMENSION
      IFLG=-1
      IF(MEX1.LE.B)GO TO 14Ø
C
C BELOW, FIRST DIMEN. GREATER THAN BLOCK SIZE, MULTIPLE BLOCKS
      NBPD1=2**(MEX1-B)
      NBLCNT=NBPD1
      KINC=NB
      KBINC=NBLCNT-1
      IF(JAY.EQ.KAY)NBLCNT=NBLCNT/2
      JB=JAY*NBPD1+1
      KB=KAY*NBPD1+1
C JB AND KB ARE BLOCK INDEX PAIRS CONTAINING CONJUGATE ELEMENTS
      J1=Ø
      K1=Ø
C
5Ø    NCNT=NB+1
6Ø    CALL MFRLOD(LOWER,IOF,BUFA,NB,JB,JBMAX,JBOF,IDIR,NCNT)
C LOWER BLOCK LOADED
      IF(JB.GT.JBMAX)IEXPND=-1
      J2=J1+IOF
      JB=JB+1
      J3=Ø
      K3=Ø
      NEWBLK=NCNT-NB
      IF(IFLG.GE.Ø)GO TO 8Ø
C
7Ø    CALL MFRLOD(LUPPR,IOF,BUFA,NB,KB,JBMAX,JBOF,IDIR,IFLG)
C UPPER BLOCK LOADED
      IFLG=IFLG+1
      K2=K1+IOF
      KB=KB+KBINC
C FIRST TIME, UPPER BLOCK STEPS HIGH, FOLLOWING STEPS SMALL NEGATIVE
      KBINC=-1
C
8Ø    J=J2+J3
      K=K2+K3
C J AND K INDEX CONJUGATE-SYMMETRIC PAIRS IN CORE
      JJ=J+NBDB
      KK=K+NBDB
      IF(IDIR.GT.Ø)GO TO 18Ø
C
C BELOW, UNSCRAMBLING FOR REAL-TO-COMPLEX FFT
      TEMP=(BUFA(J)+CONJG(BUFA(K)))*Ø.5
      BTEM=BUFA(K)-CONJG(BUFA(J))
      BTEM=(CMPLX(AIMAG(BTEM),REAL(BTEM)))*Ø.5*W
      ATEM=TEMP+BTEM
      BTEM=TEMP-BTEM
      BUFA(J)=ATEM
      IF(IEXPND.GT.Ø)BUFA(JJ)=BTEM
      IF(IWFG.EQ.Ø)GO TO 15Ø
      BUFA(K)=CONJG(BTEM)
      IF(IEXPND.GT.Ø)BUFA(KK)=CONJG(ATEM)
C
9Ø    J3=J3+1
      K3=KINC-J3
C IN-CORE INDEX PAIRS STEPPED IN OPPOSING DIRECTIONS
      IF(IDIM.NE.NDIM)GO TO 95
      IWFG=1
      W=W*D
C RECURSIVE MODIFICATION OF W IF UNIDIMEN. TRANSFORM
95    NCNT=NCNT-1
      IF(NCNT.LE.Ø)GO TO 1ØØ
C ENTER RECURSION ROUTINE IF OPERATION COMPLETE IN CURRENT DIMEN
      IF(J3.EQ.1)GO TO 7Ø
      IF(NCNT.GT.NEWBLK)GO TO 8Ø
C END OF INNER LOOP (NOTE SPECIAL CASE WHEN J3=1 ABOVE)
C
C BELOW, MAY REQUIRE TO READ NEW BLOCKS
      NBLCNT=NBLCNT-1
      IF(NBLCNT.GT.Ø)GO TO 5Ø
      IF(JAY.EQ.KAY)GO TO 6Ø
C JAY.EQ.KAY NEEDS SYMMETRICAL MIDDLE, OTHERWISE CURRENT DIMEN COMPLT
C
C 1ØØ BELOW, RECURSION TO COMPUTE MULTIDIMEN. CONJUGATE-SYMMETRY
```

```
1ØØ   JAYA(IDIM)=Ø
      KAYA(IDIM)=Ø
      IDIM=IDIM+1
      IF(IDIM.GT.NDIM)GO TO 12Ø
      NUMD=2**MEXA(IDIM)
      IF(NUMD.LE.1)GO TO 12Ø
      IF(IDIM.NE.NDIM)GO TO 1Ø5
      IWFG=1
      W=W*D
C RECURSIVE MODIFICATION OF W IF MULTIDIMEN. FFT
1Ø5   IF(JAYA(IDIM).EQ.Ø)GO TO 11Ø
      IF((JWKA(IDIM)*NUMD+JAYA(IDIM)).EQ.
     X  (KWKA(IDIM)*NUMD+KAYA(IDIM)))GO TO 1ØØ
      IF(KAYA(IDIM).EQ.1)GO TO 1ØØ
C
11Ø   JAYA(IDIM)=JAYA(IDIM)+1
      KAYA(IDIM)=NUMD-JAYA(IDIM)
C RECURSIVE STEPPING OF MULTIDIMEN. CONJUGATE-SYMMETRIC INDEX PAIRS
      GO TO 2Ø
C
C BELOW, OPERATION COMPLETE, TIDY UP AND RETURN FROM SUBROUTINE
12Ø   DO 13Ø IAREA=1,2
      CALL MFRLOD(IAREA,IOF,BUFA,NB,LCLR,JBMAX,JBOF,IDIR,IFLG)
C DUMMY CALL TO MFRLOD TO WRITE ANY UNWRITTEN BLOCKS TO MASS STORE
13Ø   CONTINUE
      RETURN
C RETURN FROM SUBROUTINE
C
C
C 14Ø BELOW, FIRST DIMEN. LESS THAN BLOCK SIZE, INDEX PAIRS ALL IN-CORE
14Ø   NUMD1=2**MEX1
      NBPD1=NB/NUMD1
      NCNT=NUMD1
      KINC=NCNT
      KBINC=Ø
      IF(JAY.EQ.KAY)NCNT=NCNT/2+1
      JB=JAY/NBPD1+1
      KB=KAY/NBPD1+1
C JB AND KB ARE BLOCK INDEX PAIRS CONTAINING CONJUGATE ELEMENTS
      J1=(JAY-(JB-1)*NBPD1)*NUMD1
      K1=(KAY-(KB-1)*NBPD1)*NUMD1
      GO TO 6Ø
C
C 15Ø BELOW, UNSCRAMBLING WITH WØ (IWFG=Ø) MUST BE TREATED DIFFERENTLY
15Ø   IF(IEXPND.LT.Ø)GO TO 16Ø
C BELOW, ARRAY EXPANSION (EITHER IPAK=+1 OR IPAK=Ø AND STILL REDUNDANT)
      BUFA(K)=CONJG(ATEM)
      BUFA(KK)=CONJG(BTEM)
      GO TO 9Ø
C 16Ø BELOW, NO ARRAY EXPANSION (EITHER IPAK=-1 OR IPAK=Ø NOT REDUNDANT)
16Ø   IF(J.EQ.K)GO TO 17Ø
      BUFA(K)=CONJG(BTEM)
      GO TO 9Ø
C 17Ø BELOW, SPECIAL CASE IF IPAK=-1 AND ELEMENTS ARE SAME
17Ø   BUFA(J)=CMPLX(REAL(ATEM),REAL(BTEM))
      GO TO 9Ø
C
C 18Ø BELOW, SCRAMBLING FOR COMPLEX-TO-REAL FFT
18Ø   IF(IWFG.EQ.Ø)GO TO 2ØØ
      BTEM=CONJG(BUFA(K))
19Ø   ATEM=(BUFA(J)+BTEM)
      BTEM=(BUFA(J)-BTEM)*W
      BTEM=CMPLX(AIMAG(BTEM),REAL(BTEM))
      BUFA(J)=ATEM-CONJG(BTEM)
      BUFA(K)=CONJG(ATEM)+BTEM
      GO TO 9Ø
C
C 2ØØ BELOW, SCRAMBLING WITH WØ (IWFG=Ø) MUST BE TREATED DIFFERENTLY
2ØØ   IF(IEXPND.LT.Ø)GO TO 21Ø
      BTEM=BUFA(JJ)
      GO TO 19Ø
C
C 21Ø BELOW, NO REDUNDANCY (EITHER IPAK=-1 OR IPAK=Ø OR 1 NOT REDUND)
21Ø   IF(J.EQ.K)GO TO 22Ø
      BTEM=CONJG(BUFA(K))
```

```
      GO TO 19Ø
C
C 22Ø BELOW, SPECIAL CASE IF IPAK=-1 AND ELEMENTS ARE SAME
22Ø   BTEM=CMPLX(AIMAG(BUFA(J)),Ø.)
      BUFA(J)=CMPLX(REAL(BUFA(J)),Ø.)
      GO TO 19Ø
C
      END


      SUBROUTINE MFRLOD(IAREA,IOF,BUFA,NB,JB,JBMAX,JBOF,IDIR,NCNT)
C
C LOADS, UNLOADS CORE STORE ARRAY BUFA, FOR REAL FFT UNSCRAMBLING ROUTINE
C  BLOCK SIZE NB CMPLX, BLOCK NUMBER JB (JB=-1 DOES FINAL TIDY O/P)
C  JBMAX IS MAX BLOCK INDX FOR EXPANSN, JBOF OFFSET TO EXPANDING BLOCKS
C  IDIR=-1 DIRECTION REAL/CMPLX, +1 CMPLX/REAL
C IAREA=1 (LOWER) OR 2 (UPPER) OF TWO AREAS IN LOGICAL UNSCRAMBLING
C  NOTE THAT BLOCK NORMALLY PHYSICALLY LOADED IN THESE AREAS,
C  BUT, IF BLOCK ALREADY RESIDENT, MAY BE IN DIFFERENT AREA, SO
C IOF RETURNED AS ACTUAL OFFSET IN BUFA TO LOADED BLOCK.
C  USES (2**B)*4 CMPLX IN BUFA, 'LOWER', 'UPPER' PLUS EXPANSION AREAS
C  RETURNS IOF AS BUFFER OFFSET TO AREA (MAY NOT BE SAME, IF BLOCK RESIDENT)
C
C NCNT IS COUNT OF ELEMENTS TO BE ACCESSED IN THIS LOAD, TO ALLOW
C  NOTE TO BE TAKEN OF ANY PARTLY FILLED BLOCKS DURING EXPANSION,
C  PREVENTING THE READING OF 'NON-EXISTENT' BLOCKS.
C LISTS JBPFA(), NCPFA() OF SIZE NPARF HOLD THIS INFORMATION, DEFAULTS
C  TO ALL-READ IF EXCEEDED, BUT INCREASE NPARF ETC, IF PROBLEM.
C
      COMPLEX BUFA(1)
      INTEGER JBAREA(2),NCAREA(2),JBPFA(5),NCPFA(5)
      DATA JBAREA(1)/-1/,JBAREA(2)/-1/,NPARF/5/
C
      IF(JBAREA(1).GE.Ø)GO TO 2Ø
C JBAREA() HOLDS INDEX OF BLOCK LOADED IN AREA 1 OR 2 (FIRST TIME -1 BELOW)
C
      IEXIST=-1
      DO 1Ø I=1,NPARF
1Ø    JBPFA(I)=-1
C PRE-CLEARS PARTLY FILLED BLOCK LIST (ONCE BLOCK FILLED, ALSO CLEARED)
C
2Ø    NBDB=NB*2
      IF(JB.LT.Ø)GO TO 5Ø
      IF(IAREA.EQ.2)GO TO 3Ø
      NCLOW=NCNT
      IF(MOD(NCNT,2).NE.Ø)NCLOW=(NCLOW-1)*2
3Ø    NCHLD=NCLOW
      IF(IAREA.EQ.2)NCHLD=NCLOW-1
      IF(NCNT.LT.Ø)NCHLD=1
C NCHLD IS THE NUMBER OF ELEMENTS TO BE ACCESSED IN CURRENT READ
C
      DO 4Ø I=1,2
      IF(JB.EQ.JBAREA(I))GO TO 14Ø
4Ø    CONTINUE
C TEST DONE TO SEE IF REQUIRED BLOCK ALREADY IN CORE (TRIVIAL IF SO)
C
C OTHERWISE BELOW, FIRST WRITE OUT RESIDENT BLOCK, THEN READ IN NEW
5Ø    IOF=(IAREA-1)*NB+1
C IOF IS BASE OFFSET OF CORE AREA WHERE BLOCK IS TO BE FOUND
      IOFDB=IOF+NBDB
      IF(JBAREA(IAREA).LT.Ø)GO TO 9Ø
      CALL MFWRIT(BUFA(IOF),NBDB,JBAREA(IAREA))
C WRITE OUT BLOCK BEFORE READING NEW BLOCK
      IF(IDIR.GT.Ø)GO TO 9Ø
      IF(JBAREA(IAREA).GT.JBMAX)GO TO 9Ø
      IF(NCAREA(IAREA).GE.NB)GO TO 8Ø
C
C BELOW, IF LAST BLOCK ONLY PART-FILLED, INDEX, ELMTS ACCESSED NOTED
      DO 6Ø I=1,NPARF
      IF(JBPFA(I).LT.Ø)GO TO 7Ø
6Ø    CONTINUE
C NO ROOM IN LISTS, DEFAULTS TO ALL READ
      I=NPARF
      IEXIST=I
7Ø    JBPFA(I)=JBAREA(IAREA)
      RDM1=DIMA(1)
```

```
      IF(IRMF.GE.Ø)RDM1=RDM1*2.
      TDM1=FSIZ/RDM1
      NCPFA(I)=NCAREA(IAREA)
C
8Ø    CALL MFWRIT(BUFA(IOFDB),NBDB,JBOF+JBAREA(IAREA))
C SIMILARLY, WRITE OUT BLOCK PAIR IF EXPANDING
C
C BELOW, READ BLOCK NOTING BLOCK INDX (READ EXPANDED, IF PART FILLED)
9Ø    JBAREA(IAREA)=JB
      IF(JB.LT.Ø)GO TO 13Ø
      CALL MFREAD(BUFA(IOF),NBDB,JB)
C READ REQUIRED BLOCK AND NOTE ACCESS COUNT
      NCAREA(IAREA)=NCHLD
      IF(JB.GT.JBMAX)GO TO 13Ø
      IF(IDIR.GT.Ø)GO TO 12Ø
C
C BELOW, EXPANSION - DOES BLOCK EXIST TO READ
      DO 1ØØ I=1,NPARF
      IF(JB.EQ.JBPFA(I))GO TO 11Ø
1ØØ   CONTINUE
      IF(IEXIST.LT.Ø)GO TO 13Ø
11Ø   JBPFA(I)=-1
      NCAREA(IAREA)=NCPFA(I)+NCHLD
C IF BLOCK TO BE READ WAS ONLY PART FILLED, THEN IT EXISTS TO READ
12Ø   CALL MFREAD(BUFA(IOFDB),NBDB,JBOF+JB)
C READ EXPANDED BLOCK IF REQUIRED
C
13Ø   RETURN
C RETURN FROM SUBROUTINE
C
C BELOW, TRIVIAL CASE - BLOCK ALREADY LOADED
14Ø   IOF=(I-1)*NB+1
C IOF IS BASE OFFSET OF CORE AREA WHERE BLOCK IS TO BE FOUND
      IF(I.NE.IAREA)GO TO 15Ø
      NCAREA(I)=NCAREA(I)+NCHLD
C INCREASE ACCESS COUNT IF CURRENT IAREA MATCHES ORIGINAL IAREA
15Ø   RETURN
C
      END


      FUNCTION MFPAR(IRMF,ICOMP)
C
C  HELPER ROUTINE TO CROSS-COMPUTE MASS STORE FFT FILE PARAMETERS
C  PARAMETERS ARE HELD AND COMPUTED IN 3 COMMON AREAS (SEE BELOW)
C  MFPAR RETURNS Ø NORMALLY, -1 IF NOT ALL MFINT CORRECT, +1 IBEX ERROR
C
C  COMMON/MFARG/ HOLDS ARGUMENTS AS USED IN FFT CALLS, AS FOLLOWS:
C  VARIABLE NAMES HAVE SAME MEANING AS COMMENTS, SUBROUTINE RMFFT
C     MEXA() HOLDS EXPONENTS FOR UP TO 4 DIMENSIONS (R/T ZEROS EXCESS)
C     NDIM NUM DIMENS, IBEX,ICEX BLOCK AND CORE EXPONS, IPAK RMFFT PACKING
C     ISGN,IDIR,SCAL ARE IGNORED HERE, BUT INCLUDED FOR COMPLETENESS
C
C  COMMON/MFVAL/,/MFINT/ RETURN COMPUTED VALUES, USEFUL FOR FILE ACCESS
C     NDMA(4),DIMA(4) HOLD DIMENSION SIZES CORRESPONDING TO MEXA()
C     (EG. NDMA(1)=DIMA(1)=2.**MEXA(1), ETC. AND =1. BEYOND NDIM)
C
C     NTD1,TDM1  IS CURRENT TOTAL NUM OF 'RECDS' OF SIZE NRD1,RDM1
C     NRD1,RDM1  IS NUM OF REALS IN CURRENT FIRST DIMENSION
C     (USEFUL FOR ACCESSING DATA BY MFREAD/MFWRIT, NRD1,RDM1 REALS,
C      ASSUMING THAT MFREAD/MFWRIT CAN HANDLE 'RECDS' OF DIFFERENT SIZES)
C
C     NFBK,FBLK  IS MAXIMUM FILE SIZE OF 'RECDS' OF SIZE NRBK,RBLK
C     NTBK,TBLK  IS CURRENT TOTAL NUM OF 'RECDS' OF SIZE NRBK,RBLK
C     NRBK,RBLK  IS NUM OF REALS IN FFT WORKING BLOCK (2**IBEX REALS)
C     (GIVES MAX AND CURRENT FILE SIZE AND ACCESS BY FFT ROUTINES,
C   NFBK.GT.NTBK ONLY WITH PACKED REAL DATA WHEN EXPANDING, IPAK=Ø OR 1)
C
C     NRCR,RCOR  IS NUM OF REALS IN FFT WORKING CORE  (2**ICEX REALS)
C     NSZE=SIZE=2.**M, WHICH IS THE EFFECTIVE TOTAL SIZE OF TRANSFORM,
C        WHERE M IS SUM TO NDIM OF MEXA() (SEE RMFFT COMMENTS)
C
C  NOTE THAT ALL /MFARG/ ARE INTGS (EXCEPT SCAL), ALL /MFVAL/ REALS
C  (/MFINT/ IS INTEGER CONVERSION OF /MFVAL/, ANY VALUE OF MFINT
C   IS SET -1 IF TOO LARGE, BY FIXMAX, AND MFPAR RETURNED -1 AS FLAG,
C   FIXMAX SET BY DATA STATEMENT TO 32767. HERE, BUT ALTER TO SUIT)
C
```

```
C  NOTE WELL THAT TYPE COMPLEX DATA MUST EXIST AS ALTERNATING
C  REAL/IMAG/REAL/IMAG... ELEMENTS, BOTH IN MASS STORE AND IN FORTRAN
C  WORKING ARRAY BUFA;  IN THIS FFT, DIFFERENT SUBROUTINES WILL SET
C ROUTINE ARGUMENTS HAVE THE FOLLOWING EFFECT:
C  IRMF=-1, DATA IS PACKED REAL, +1 DATA IS COMPLEX, ROUTINE RMFFT,
C  IRMF=Ø,  DATA IS COMPLEX, ROUTINE CMFFT
C
C  ICOMP=Ø, COMPUTES VALUES IN /MFVAL/ FROM VALUES GIVEN IN /MFARG/
C  ICOMP=1 , REVERSE COMPUTES EXPONENTS IN /MFARG/ FROM /MFVAL/
C     (DIMA(),RBLK,RCOR GIVEN INSTEAD OF MEXA(),IBEX,ICEX)
C
C  NOTE, ROUTINE FORCES ICEX, IBEX TO CORRECT RANGE, MFPAR=+1 IF CANNOT
C
      COMMON/MFARG/MEXA(4),NDIM,ISGN,IDIR,SCAL,IBEX,ICEX,IPAK
      COMMON/MFVAL/DIMA(4),TDM1,RDM1,FBLK,TBLK,RBLK,RCOR,SIZE
      COMMON/MFINT/NDMA(4),NTD1,NRD1,NFBK,NTBK,NRBK,NRCR,NSZE
      REAL VAL(11)
      INTEGER INT(11)
      EQUIVALENCE (VAL(1),DIMA(1)),(INT(1),NDMA(1))
C
      DATA FIXMAX/32767./,NMAX/4/
C
      MFPAR=Ø
      IF(ICOMP.EQ.Ø)GO TO 2Ø
      ALG2=ALOG(2.)
      IBEX=IFIX(ALOG(RBLK)/ALG2+Ø.5)
      ICEX=IFIX(ALOG(RCOR)/ALG2+Ø.5)
C
      DO 1Ø I=1,NDIM
1Ø    MEXA(I)=IFIX(ALOG(DIMA(I))/ALG2+Ø.5)
C
2Ø    M=Ø
      DO 3Ø I=1,NMAX
      IF(I.GT.NDIM)MEXA(I)=Ø
      M=M+MEXA(I)
3Ø    DIMA(I)=2.**MEXA(I)
      SIZE=2.**M
C
      IF(IRMF.EQ.Ø)GO TO 9Ø
      IF(ICEX.GT.M)ICEX=M
      IF(IBEX.GT.ICEX-2)IBEX=ICEX-2
      IF(IBEX.LT.2)MFPAR=1
C FORCES ICEX.NGT.M AND IBEX.NGT.ICEX-2, OR MFPAR=1 (IRMF=+/- 1)
C
4Ø    RBLK=2.**IBEX
      RCOR=2.**ICEX
C
      FADD=SIZE
      IF(IRMF.EQ.Ø.OR.IPAK.GT.Ø)GO TO 5Ø
C FADD IS ADDITIONAL FILE SIZE IN REALS, 'SIZE' IF CMFFT OR IPAK=1
C
      FADD=Ø.
      IF(IPAK.LT.Ø)GO TO 5Ø
C IPAK=-1 REQUIRES NO FILE EXPANSION
C
      IDIM=NDIM
      IF(IRMF.LT.Ø)IDIM=1
      FADD=SIZE*2./DIMA(IDIM)
C FADD COMPUTED FOR PARTICULAR CASE OF IPAK=Ø, WHEN COMPLX
C
5Ø    FSIZ=SIZE+FADD
      ITEM=IFIX(FSIZ/RBLK+Ø.5)
      IF(FLOAT(ITEM)*RBLK+Ø.5 .LT.FSIZ)ITEM=ITEM+1
      FBLK=FLOAT(ITEM)
C FBLK IS MAXIMUM NUMBER OF 'RECDS', SIZE RBLK, POSSIBLE
      TBLK=FBLK
      IF(IRMF.GE.Ø)GO TO 6Ø
C GENERALLY TBLK=FBLK, BUT FOR PACKED REAL NOT SO, BELOW
      FSIZ=SIZE
      TBLK=FSIZ/RBLK
C
6Ø    TDM1=1.
      RDM1=FSIZ
      IF(NDIM.EQ.1)GO TO 7Ø
C JOB COMPLETED IF NDIM=1
C
```

```
C OTHERWISE COMPUTE TDM1 AS NUMBER OF 'RECDS', SIZE RDM1 REALS
C
7Ø    DO 8Ø I=1,11
      INT(I)=-1
      IF(VAL(I).LE.FIXMAX)INT(I)=IFIX(VAL(I)+Ø.5)
      IF(INT(I).LT.Ø.AND.MFPAR.EQ.Ø)MFPAR=-1
8Ø    CONTINUE
C CONVERT VALUES IN /MFVAL/ TO INTEGERS IN /MFINT/ (-1 IF TOO LARGE)
      RETURN
C
9Ø    IF(ICEX.GT.M+1)ICEX=M+1
      IF(IBEX.GT.ICEX-1)IBEX=ICEX-1
      IF(IBEX.LT.1)MFPAR=1
      GO TO 4Ø
C  FORCES ICEX.NGT.M+1 AND IBEX.NGT.ICEX-1, OR MFPAR=1 (IRMF=Ø)
C
      END


      SUBROUTINE DMPERM (MEXA,NDIM,NSHFT,IREX, BUFA,IBEX,ICEX)
C
C SHIFTS ORDER OF DIMENSIONS OF REAL OR COMPLEX MASS STORE ARRAY
C NOTE, THIS IS NOT USED BY FFT SUBRTNS BUT IS INCLUDED FOR COMPLETENESS
C  (FRASER, ACM TOMS - 1978/79, AND J.ACM, V.23,N.2, APRIL 76, PP. 298-3Ø9)
C
C MEXA(J) LIST OF DIMENSION SIZE EXPONS (BASE 2), ADJACENT VARIABLES FIRST
C NDIM IS NUMBER OF EXPONENTS IN LIST AND THUS THE NUMBER OF DIMENSIONS
C  SUM TO NDIM: MEXA(J)=M, WHERE 2**M IS SIZE OF MASS STORE ARRAY (SEE BELOW)
C NSHFT IS DIMENSION SHIFT COUNT, THUS:
C  NSHFT=Ø,  NO SHIFT OR CHANGE OCCURS
C  NSHFT=1,2 ETC., FIRST TO NEXT DIMENSION, CIRC NSHFT PLACE SHIFT (MOD NDIM)
C  NSHFT=-1, REVERSES THE ORDER OF DIMENSIONS
C IREX=Ø REAL, 1 COMPLEX (THAT IS, MOVEMENT GROUP IS 2**IREX REALS,
C    AND TOTAL MASS STORE SIZE IS 2**(M+IREX) REAL ELEMENTS)
C
      REAL BUFA(1)
      INTEGER MEXA(1)
C
      NS=MOD(NSHFT,NDIM)
      IF(NS.EQ.Ø)RETURN
      M=MFSUM(MEXA,NDIM,-1)+IREX
C FINDS M TOTAL AND REVERSES MEXA LIST
      CALL MFSORT(BUFA,IBEX,ICEX,IREX,M,M)
C INITIAL OVERALL BIT-REVERSAL M BITS ABOVE IREX BITS
      IF(NSHFT.LT.Ø)GO TO 1Ø
C
C BELOW, REVERSAL OF TWO PARTS, TO FORM REQUIRED SHIFT
      IH=MFSUM(MEXA,NS,-1)+IREX
      CALL MFSORT(BUFA,IBEX,ICEX,IREX,IH,M)
C REVERSE LOWER PART OF MEXA LIST AND LOWER PART OF ARRAY BITS
      CALL MFSORT(BUFA,IBEX,ICEX,IH,M,M)
C SEPARATELY REVERSE UPPER PART OF ARRAY BITS
      IH=MFSUM(MEXA(NSHFT+1),NDIM-NS,-1)
C REVERSE UPPER PART OF MEXA LIST
      RETURN
C RETURN FROM SUBROUTINE AFTER CYCLIC SHIFTS
C
C BELOW, SEPARATELY REVERSE OVER EACH DIMENSION (DIMEN REVERSAL)
1Ø    IH=IREX
      DO 2Ø J=1,NDIM
      IG=IH
      IH=IH+MEXA(J)
2Ø    CALL MFSORT(BUFA,IBEX,ICEX,IG,IH,M)
      RETURN
C
      END


C
C THIS PROGRAM TESTS THE MASS STORE FFT BY COMPARISON WITH NAIVE DFT
C     FFT PARAMETERS MAY BE ALTERED AT WILL (SEE COMMENTS)
C MASS STORE IS SIMULATED BY FORTRAN ARRAYS (SEE DUMMY I/O SUBROUTINES)
C PRINTING MAY BE COPIOUS, OR ONLY MAX DIFFERENCES (SEE COMMENTS)
C TEST OK IF MAX DIFFERENCES ARE NEAR ORDER OF MACHINE ROUND-OFF
C
```

```
C  DIFFERENT TYPE (REAL OR COMPLEX) FOR ARRAY BUFA.
C
      COMMON/MFARG/MEXA(4),NDIM,ISGN,IDIR,SCAL,IBEX,ICEX,IPAK
      COMMON/MFVAL/DIMA(4),TDM1,RDM1,FBLK,TBLK,RBLK,RCOR,SIZE
      COMMON/MFINT/NDMA(4),NTD1,NRD1,NFBK,NTBK,NRBK,NRCR,NSZE
C
C COMMON AREAS /MFARG/,/MFVAL/,/MFINT/ USEFUL FOR RUNNING MASS STORE FFT
C  /MFARG/ HOLDS ARGUMENTS USED IN FFT CALLES, MOSTLY EXPONENTS
C    HELPER ROUTINE MFPAR COMPUTES VALUES FROM /MFARG/ INTO
C  /MFVAL/ (REALS AS SOME LARGE), /MFINT/ (INTEGER EQUIVALENTS IF POSS)
C    OR CAN REVERSE-COMPUTE SOME /MFARG/ FROM /MFVAL/
C       SEE COMMENTS, ROUTINE MFPAR.
C
      COMMON/MASS/RMAS(1Ø24)
      COMPLEX ANAIV(512),BNAIV(512),CBUFA(512),CDIF
      REAL RANDA(1Ø24),BUFA(1Ø24)
      EQUIVALENCE (CBUFA(1),BUFA(1))
C
C COMMON/MASS/RMAS() USED BY ROUTINES MFREAD/MFWRIT TO SIMULATE MASS STORE
C  ANAIV(), BNAIV()  HOLD RESULT OF NAIVE DFT FOR COMPARISON
C  BUFA()=CBUFA()    IS WORKING AREA IN CORE STORE FOR FFT AND PROGRAM
C  RANDA  HOLDS PSEUDO RANDOM DATA USED IN TEST
C
      LP=5
      IPRINT=Ø
C LP IS PRINTER LOGICAL UNIT, IPRINT=+1 FOR COPIOUS PRINT,
C  IPRINT=Ø FOR MAX DIFFERENCES ONLY, -1 OVERALL MAX DIFFERENCE ONLY
C
      M=5
C M SETS THE OVERALL ARRAY SIZE FOR AUTO IBEX,ICEX,MEXA,NDIM STEPPING
C  FIXED VALUES CAN BE USED (SEE COMMENTS BELOW DO 1ØØ STATEMENTS)
C
      IRMF=-1
C IRMF=-1 REAL ROUTINE RMFFT TEST, Ø COMPLEX ROUTINE CMFFT TEST
C
      ISGN=-1
      IDIR=-1
      IPAK=1
C FFT ARGS, ISGN=+/- 1, IDIR=-1 (RMFFT), IDIR=1 OR Ø (CMFFT)
C IPAK=1 OR Ø (RMFFT), -1 GIVES APPARENT FAILURES DUE TO SQUEESED RESULT
C
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      IF(IPRINT.LE.Ø)WRITE(LP,91Ø)IPRINT,IRMF
91Ø   FORMAT(31H1MASS STORE FFT TEST - IPRINT =,I3,
     X  36H (1=COPIOUS,Ø=MAX DIFFS,-1=OVERALL),,
     X  9H   IRMF =,I3,28H (Ø=CMFFT TEST,1=RMFFT TEST) /)
      DIFMG=Ø.
C
C BELOW, DO 1ØØ COMPUTES ALL POSSIBLE MEXA FOR 1,2 AND 3 DIMENSIONS
      DO 1ØØ NDIM=1,3
      M2M=M-NDIM+1
      IF(NDIM.EQ.1)M2M=1
      DO 1ØØ M2=1,M2M
      M3M=M2M-M2+1
      IF(NDIM.NE.3)M3M=1
      DO 1ØØ M3=1,M3M
      IF(NDIM.EQ.1)MEXA(1)=M
      IF(NDIM.EQ.2)MEXA(1)=M-M2
      IF(NDIM.EQ.3)MEXA(1)=M-M2-M3
      MEXA(2)=M2
      MEXA(3)=M3
C REPLACE DO 1ØØ SET BY FIXED MEXA LIST, IF DESIRED
C
      IBEX=2
      ICEX=4
C DUMMY IBEX, ICEX SO NO ERROR IN MFPAR BELOW
      IERR=MFPAR(IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
C CALL HELPER ROUTINE TO COMPUTE SIZES USED BY NAIVE DFT SUBRTN
C
      M=MFSUM(MEXA,NDIM,99)
      AR=RANMF(1)
C RESET RANDOM NUMBER GENERATOR AND COMPUTE M IN CASE NOT GIVEN
      DO 1Ø J=1,NSZE
      RANDA(J)=RANMF(-1)
1Ø    ANAIV(J)=CMPLX(RANDA(J),Ø.)
```

```
C LOAD RANDOM NUMBERS FOR FFT AND FOR NAIVE SUBROUTINE
C
      CALL NAIVE(ANAIV,BNAIV,NDMA(1),NDMA(2),NDMA(3),ISGN)
C NAIVE DFT SUBROUTINE CALLED TO COMPUTE 'SLOW' FOURIER TRANSFORM
C
C
      IF(IRMF.EQ.Ø)GO TO 2Ø
C BELOW, SET LIMTS FOR STEPPING IBEX, ICEX, WHEN CALLING RMFFT
      IBEXL=2
      ICEXL=4
      ICEXM=M
      GO TO 3Ø
C
C SETS DIFFERENT LIMITS FOR IBEX, ICEX FOR CMFFT BELOW (IRMF.EQ.Ø)
2Ø    IBEXL=1
      ICEXL=2
      ICEXM=M+1
C
3Ø    IF(ICEXL.GT.ICEXM)GO TO 1ØØØ
      DO 1ØØ ICEX=ICEXL,ICEXM
      IBEXM=ICEX-2
      IF(IRMF.EQ.Ø)IBEXM=ICEX-1
      DO 1ØØ IBEX=IBEXL,IBEXM
C IBEX AND ICEX COMPUTED; REPLACE DO 1ØØ ABOVE BY FIXED IBEX,ICEX IF REQUD.
C
      IERR=MFPAR(IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
      NCNT=NRD1
C HELPER ROUTN, NTD1 TOTAL NUMB OF NRD1 REALS IN FIRST DIMENSION
C   (NCNT IS NUMBER OF REAL ELMTS IN FIRST DIMENSION)
      SCAL=1.
      IF(IRMF.EQ.Ø)GO TO 5Ø
C SWITCH FOR COMPLEX ROUTINE CMFFT AT 5Ø, REAL RMFFT BELOW
C
      DO 4Ø JB=1,NTD1
      K=(JB-1)*NCNT
      DO 35 I=1,NCNT
      J=K+I
35    BUFA(I)=RANDA(J)
      CALL MFWRIT(BUFA,NRD1,JB)
4Ø    CONTINUE
C LOAD RANDOM NUMBERS IN REAL ARRAY IN 'RECDS' OF FIRST DIMEN LENGTH
C  (NOTE, THIS REQUIRES MFREAD/MFWRIT TO BE ABLE TO ACCEPT 'RECORDS'
C   OF DIFFERENT LENGTH; OTHERWISE MUST USE NTBK AND NRBK HERE)
C
      CALL RMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX, IPAK)
C REAL MASS STORE ROUTINE RMFFT TO TRANSFORM ARRAY TO COMPLEX RESULT
      GO TO 7Ø
C
C BELOW, USE COMPLEX ROUTINE CMFFT,  NCNT NUM OF CMPLX ELMTS IN FIRST DIMEN
5Ø    NCNT=NCNT/2
      DO 6Ø JB=1,NTD1
      K=(JB-1)*NCNT
      DO 55 I=1,NCNT
      J=K+I
55    CBUFA(I)=CMPLX(RANDA(J),Ø.)
      CALL MFWRIT(BUFA,NRD1,JB)
6Ø    CONTINUE
C LOAD REAL VALUES IN CMPLX ARRAY IN 'RECDS' OF FIRST DIMEN LENGTH
C  (NOTE, THIS REQUIRES MFREAD/MFWRIT TO BE ABLE TO ACCEPT 'RECORDS'
C   OF DIFFERENT LENGTH; OTHERWISE MUST USE NTBK AND NRBK HERE)
C
      CALL CMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX)
C COMPLEX MASS STORE ROUTINE CMFFT TO TRANSFORM ARRAY TO COMPLEX RESULT
C
7Ø    IF(IPRINT.LE.Ø)GO TO 75
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      WRITE(LP,91Ø)IPRINT,IRMF
      WRITE(LP,92Ø)NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
92Ø   FORMAT(8H  NDIM =,I3,8H  ISGN =,I3,8H  IDIR =,I3,
     X  8H  IBEX =,I3,8H  ICEX =,I3,8H  IPAK =,I3,
     X  1ØH  MEXA() =,3I5)
      WRITE(LP,93Ø)
93Ø   FORMAT(8H    INDEX,12X,3HFFT,24X,5HNAIVE,24X,4HDIFF)
C
```

```
75    IERR=MFPAR(-IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
      NCNT=NRD1/2
C HELPER ROUTN, NTD1 TOTAL NUMB OF NRD1 REALS IN FIRST DIMENSION
C  (NOTE IRMF=+1 FOR DATA IN COMPLEX STATE, NCNT ELMTS)
C
C BELOW, COMPARES FFT COMPLEX RESULT WITH NAIVE RESULT
      DIFM=Ø.
      DO 8Ø JB=1,NTD1
      K=(JB-1)*NCNT
      CALL MFREAD(BUFA,NRD1,JB)
C READ 'RECDS' OF FIRST DIMEN LENGTH (RECOMPUTED ABOVE BY MFPAR)
C  (NOTE, THIS REQUIRES MFREAD/MFWRIT TO BE ABLE TO ACCEPT 'RECORDS'
C   OF DIFFERENT LENGTH; OTHERWISE MUST USE NTBK AND NRBK HERE)
C
      DO 8Ø I=1,NCNT
      J=K+I
      INDEX=J-1
      IF(IDIR.NE.Ø)CDIF=CBUFA(I)-BNAIV(J)
      IF(IDIR.EQ.Ø)CDIF=CBUFA(I)-ANAIV(J)
      DIF=ABS(REAL(CDIF))
      IF(DIF.GT.DIFM)DIFM=DIF
      DIF=ABS(AIMAG(CDIF))
      IF(DIF.GT.DIFM)DIFM=DIF
      IF(IPRINT.LE.Ø)GO TO 8Ø
      IF(IDIR.NE.Ø)WRITE(LP,94Ø)INDEX,CBUFA(I),BNAIV(J),CDIF
      IF(IDIR.EQ.Ø)WRITE(LP,94Ø)INDEX,CBUFA(I),ANAIV(J),CDIF
94Ø   FORMAT(1X,I5,3(2X,2E13.4))
8Ø    CONTINUE
C BELOW, PRINT INTERMEDIATE DIFFERENCES
C
      IF(IPRINT.GE.Ø)WRITE(LP,95Ø)DIFM,NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
95Ø   FORMAT(1ØH MAX DIFF ,E11.4,1X,
     X  8H  NDIM =,I3,8H  ISGN =,I3,8H  IDIR =,I3,
     X  8H  IBEX =,I3,8H  ICEX =,I3,8H  IPAK =,I3,
     X  1ØH  MEXA() =,3I5)
      IF(DIFM.GT.DIFMG)DIFMG=DIFM
C
C BELOW, INVERT ISGN AND IDIR FOR INVERSE TRANSFORM (COMPLEX-TO-REAL)
      ISGN=-ISGN
      IDIR=-IDIR
      SCAL=1./SIZE
      IF(IRMF.NE.Ø)
     X  CALL RMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX, IPAK)
C EITHER ROUTINE RMFFT INVERSE TRANSFORMS ARRAY TO PACKED REAL
C
      IF(IRMF.EQ.Ø)
     X  CALL CMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX)
C OR ROUTINE CMFFT INVERSE TRANSFORMS ARRAY
C
      IF(IPRINT.LE.Ø)GO TO 85
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      WRITE(LP,91Ø)IPRINT,IRMF
      WRITE(LP,92Ø)NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
      WRITE(LP,96Ø)
96Ø   FORMAT(8H   INDEX,4X,8HFFT/2**M,6X,5HINPUT,1ØX,4HDIFF)
C
85    IERR=MFPAR(IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
      NCNT=NRD1
      IF(IRMF.EQ.Ø)NCNT=NCNT/2
C HELPER ROUTN, NTD1 TOTAL NUMB OF NRD1 REALS IN FIRST DIMENSION
C   (NCNT IS NUMBER OF ELMTS IN FIRST DIMENSION, REAL OR CMPLX)
C
C BELOW, COMPARES FFT INVERSE RESULT WITH INITIAL RANDOM INPUT
      DIFM=Ø.
      DO 9Ø JB=1,NTD1
      K=(JB-1)*NCNT
      CALL MFREAD(BUFA,NRD1,JB)
C READ 'RECDS' OF FIRST DIMEN LENGTH (RECOMPUTED ABOVE BY MFPAR)
C  (NOTE, THIS REQUIRES MFREAD/MFWRIT TO BE ABLE TO ACCEPT 'RECORDS'
C   OF DIFFERENT LENGTH; OTHERWISE MUST USE NTBK AND NRBK HERE)
C
      DO 9Ø I=1,NCNT
```

```
      J=K+I
      INDEX=J-1
      IF(IRMF.NE.Ø)RM=BUFA(I)
      IF(IRMF.EQ.Ø)RM=REAL(CBUFA(I))
      DIF=ABS(RM-RANDA(J))
      IF(DIF.GT.DIFM)DIFM=DIF
      IF(IPRINT.GT.Ø)WRITE(LP,94Ø)INDEX,RM,RANDA(J),DIF
9Ø    CONTINUE
C PRINT INVERSE RESULTS (SHOULD BE SAME AS INITIAL RANDOM SET)
C
      IF(IPRINT.GE.Ø)WRITE(LP,95Ø)DIFM,NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
      IF(DIFM.GT.DIFMG)DIFMG=DIFM
      ISGN=-ISGN
      IDIR=-IDIR
C RESTORE ISGN AND IDIR FOR FORWARD TRANSFORM
1ØØ   CONTINUE
C
C BELOW, PRINT OVERALL MAXIMUM DIFFERENCE
      WRITE(LP,97Ø)DIFMG,M,ISGN,IDIR,IPAK
97Ø   FORMAT(18H OVERALL MAX DIFF ,E11.4,3X,
     X 48H FOR ALL MEXA(1 TO 3 DIM),IBEX,ICEX,MEXA FOR M =,I3,
     X 19H,  ISGN,IDIR(+/-) =,2I4,8H  IPAK =,I4)
      STOP
C
1ØØØ  WRITE(LP,98Ø)IERR
98Ø   FORMAT(25H IBEX FORCED TOO SMALL OR,
     X   3ØH NOT ALL /MFINT/ CORRECT, IERR,I4)
      STOP
      END


      FUNCTION RANMF(J)
C
C RANDOM NUMBER GENERATOR FOR MASS STORE FFT TEST
C
      IF(J.GE.Ø)GO TO 2Ø
C POSITIVE J CAUSES RESET OF INITIAL K
C NEGATIVE J MUST BE USED NORMALLY
C
      MODULO=2Ø48
      FLMOD=2Ø48.Ø
      DO 1Ø I=1,15
1Ø    K=MOD(5*K,MODULO)
      Z=FLOAT(K)/FLMOD
      RANMF=Z
      RETURN
C
2Ø    K=J
      RANMF=J
      RETURN
C
      END


      SUBROUTINE NAIVE(ANAIV,BNAIV,NJ,NK,NL,ISGN)
C
C NAIVE DISCRETE FOURIER TRANSFORM - 1 TO 3 DIMENSIONS
C  USED TO TEST MASS STORE FFT, INPUT ARRAY  ANAIV(NJ,NK,NL)
C RESULT RETURNED IN BOTH ARRAYS ANAIV AND BNAIV
C  ANAIV DIMENSIONS IN INITIAL ORDER, BNAIV REVERSED ORDER
C  NJ,NK,NL DIMENSIONING, ISGN SIGN OF COMPLEX EXPONENT OF FOURIER
C
      COMPLEX TEMP,ANAIV(NJ,NK,NL),BNAIV(NL,NK,NJ)
      DATA PI/3.141592653589793/
C
      PI2=PI*2.Ø
      IF(ISGN.LT.Ø)PI2=-PI2
C
      DO 2Ø JB=1,NJ
      AJB=FLOAT(JB-1)/FLOAT(NJ)
      DO 2Ø KB=1,NK
      AKB=FLOAT(KB-1)/FLOAT(NK)
      DO 2Ø LB=1,NL
      ALB=FLOAT(LB-1)/FLOAT(NL)
C
```

```
      TEMP=(Ø.,Ø.)
      DO 1Ø JA=1,NJ
      AJA=FLOAT(JA-1)*AJB
      DO 1Ø KA=1,NK
      AKA=FLOAT(KA-1)*AKB
      DO 1Ø LA=1,NL
      ALA=FLOAT(LA-1)*ALB
1Ø    TEMP=TEMP+ANAIV(JA,KA,LA)*CEXP(CMPLX(Ø.,PI2*(AJA+AKA+ALA)))
C
2Ø    BNAIV(LB,KB,JB)=TEMP
C
      DO 3Ø JA=1,NJ
      DO 3Ø KA=1,NK
      DO 3Ø LA=1,NL
3Ø    ANAIV(JA,KA,LA)=BNAIV(LA,KA,JA)
      RETURN
C
      END


      SUBROUTINE MFREAD(BUFA,NB,JB)
C
C DUMMY SUBROUTINE TO SIMULATE RANDOM ACCESS MASS STORE READ
C READ BLOCK, INDEX JB, FROM MASS STORE TO BUFA, NB REAL VALUES
C COMMON ARRAY RMAS SIMULATES MASS STORE ARRAY
C
      COMMON/MASS/RMAS(1Ø24)
      REAL BUFA(NB)
C
      IOF=(JB-1)*NB
      DO 1Ø I=1,NB
      K=IOF+I
1Ø    BUFA(I)=RMAS(K)
      RETURN
C
      END


      SUBROUTINE MFWRIT(BUFA,NB,JB)
C
C DUMMY SUBROUTINE TO SIMULATE RANDOM ACCESS MASS STORE WRITE
C WRITE BLOCK, INDEX JB, FORM BUFA TO MASS STORE, NB REAL VALUES
C COMMON ARRAY RMAS SIMULATES MASS STORE ARRAY
C
      COMMON/MASS/RMAS(1Ø24)
      REAL BUFA(NB)
C
      IOF=(JB-1)*NB
      DO 1Ø I=1,NB
      K=IOF+I
1Ø    RMAS(K)=BUFA(I)
      RETURN
C
      END


      PROGRAM MASTOM(TAPE1,INPUT,OUTPUT,TAPE6Ø=INPUT,TAPE5=OUTPUT)
C
C  CONTROL DATA 6ØØØ AND CYBER MASS STORE I/O FFT TEST PROGRAM
C
C  NOTE WELL THAT TYPE COMPLEX DATA MUST EXIST AS ALTERNATING
C  REAL/IMAG/REAL/IMAG... ELEMENTS, BOTH IN MASS STORE AND IN FORTRAN
C  WORKING ARRAY BUFA;  IN THIS FFT, DIFFERENT SUBROUTINES WILL SET
C  DIFFERENT TYPE (REAL OR COMPLEX) FOR ARRAY BUFA.
C
C
      COMMON/FFTCOM/LUN,MINDX(512)
C
C COMMON /FFTCOM/ HOLDS LOGICAL UNIT NUMBER FOR MASS STORE I/O
C ARRAY MINDX HOLDS RECORD INDICES FOR CYBER MASS STORE I/O
C
      COMMON/MFARG/MEXA(4),NDIM,ISGN,IDIR,SCAL,IBEX,ICEX,IPAK
      COMMON/MFVAL/DIMA(4),TDM1,RDM1,FBLK,TBLK,RBLK,RCOR,SIZE
      COMMON/MFINT/NDMA(4),NTD1,NRD1,NFBK,NTBK,NRBK,NRCR,NSZE
C
```

```
C COMMON AREAS /MFARG/,/MFVAL/,/MFINT/ USEFUL FOR RUNNING MASS STORE FFT
C  /MFARG/ HOLDS ARGUMENTS USED IN FFT CALLES, MOSTLY EXPONENTS
C    HELPER ROUTINE MFPAR COMPUTES VALUES FROM /MFARG/ INTO
C  /MFVAL/ (REALS AS SOME LARGE), /MFINT/ (INTEGER EQUIVALENTS IF POSS)
C    OR CAN REVERSE-COMPUTE SOME /MFARG/ FROM /MFVAL/
C       SEE COMMENTS, ROUTINE MFPAR.
C
      COMPLEX CBUFA(4Ø96)
      REAL BUFA(8192)
      EQUIVALENCE (CBUFA(1),BUFA(1))
C
C  BUFA()=CBUFA()    IS WORKING AREA IN CORE STORE FOR FFT AND PROGRAM
C
      LUN=1
C LUN IS LOGICAL UNIT FOR CYBER MASS STORE I/O
C
      LP=5
      IPRINT=Ø
C LP IS PRINTER LOGICAL UNIT, IPRINT=+1 COPIOUS, Ø MAX DIFFERENCES ONLY
C
      IRMF=-1
C IRMF=-1 REAL ROUTINE RMFFT TEST, Ø COMPLEX ROUTINE CMFFT TEST
C
      ISGN=-1
      IDIR=-1
      IPAK=1
C FFT ARGS, ISGN=+/- 1, IDIR=-1 (RMFFT), IDIR=1 OR Ø (CMFFT)
C IPAK=1, Ø OR -1 (RMFFT), NOT USED (CMFFT)
C
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      IF(IPRINT.LE.Ø)WRITE(LP,91Ø)IPRINT,IRMF
91Ø   FORMAT(31H1MASS STORE FFT TEST - IPRINT =,I3,
     X  25H (1=COPIOUS,Ø=MAX DIFFS),,
     X  9H   IRMF =,I3,28H (Ø=CMFFT TEST,1=RMFFT TEST) /)
      DIFMG=Ø.
C
      MEXA(1)=8
      MEXA(2)=6
      NDIM=2
      IBEX=9
      ICEX=13
C MORE FFT ARGS, 2**8 ROWS OF 2**6 ELMTS, 2 DIMEN,
C  FFT MASS STORE BLOCKS 2**IBEX REALS, BUFA 2**ICEX REALS
C
      IERR=MFPAR(IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
      CALL OPENMS(LUN,MINDX,NFBK+1,Ø)
C CYBER 'READMS/WRITMS' MASS STORE OPENED WITH 'NUM REC'+1=NFBK+1
C  (NOTE HELPER ROUTN MFPAR RETURNS NFBK AS MAXIMUM FILE SIZE)
C
      NCNT=NRBK
C HELPER ROUTINE, NCNT NUM OF ELMTS IN FFT BLOCK, NTBK TOTAL BLOCKS
C
      M=MFSUM(MEXA,NDIM,99)
      SCAL=1.
      VALU=Ø.
      VINC=1./SIZE
      IF(IRMF.EQ.Ø)GO TO 5Ø
C SWITCH FOR COMPLEX ROUTINE CMFFT AT 5Ø, REAL RMFFT BELOW
C
      DO 4Ø JB=1,NTBK
      DO 35 I=1,NCNT
      BUFA(I)=VALU
35    VALU=VALU+VINC
      CALL MFWRIT(BUFA,NRBK,JB)
4Ø    CONTINUE
C LOAD RAMP FUNCTN IN REAL ARRAY IN 'RECDS' OF NRBK LENGTH
C  (NOTE, CYBER MFREAD/MFWRIT (USING READMS/WRITMS) CANNOT ACCEPT 'RECDS'
C   OF DIFFERENT LENGTH; OTHERWISE COULD USE NTD1 'RECDS' OF NRD1 HERE)
C
      CALL RMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX, IPAK)
C REAL MASS STORE ROUTINE RMFFT TO TRANSFORM ARRAY TO COMPLEX RESULT
      GO TO 7Ø
C
C BELOW, USE COMPLEX ROUTINE CMFFT,  NCNT NUM OF CMPLX ELMTS IN FFT BLOCK
5Ø    NCNT=NCNT/2
```

```
      DO 6Ø JB=1,NTBK
      DO 55 I=1,NCNT
      CBUFA(I)=CMPLX(VALU,Ø.)
55    VALU=VALU+VINC
      CALL MFWRIT(BUFA,NRBK,JB)
6Ø    CONTINUE
C LOAD RAMP FUNCTN IN COMPLEX ARRAY IN 'RECDS' OF NRBK LENGTH
C  (NOTE, CYBER MFREAD/MFWRIT (USING READMS/WRITMS) CANNOT ACCEPT 'RECDS'
C   OF DIFFERENT LENGTH; OTHERWISE COULD USE NTD1 'RECDS' OF NRD1 HERE)
C
      CALL CMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX)
C COMPLEX MASS STORE ROUTINE CMFFT TO TRANSFORM ARRAY TO COMPLEX RESULT
C
7Ø    IF(IPRINT.LE.Ø)GO TO 75
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      WRITE(LP,91Ø)IPRINT,IRMF
      WRITE(LP,92Ø)NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
92Ø   FORMAT(8H  NDIM =,I3,8H  ISGN =,I3,8H  IDIR =,I3,
     X  8H  IBEX =,I3,8H  ICEX =,I3,8H  IPAK =,I3,
     X  1ØH  MEXA() =,3I5)
      WRITE(LP,93Ø)
93Ø   FORMAT(8H   INDEX,12X,3HFFT)
C
      IERR=MFPAR(-IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
      NCNT=NRBK/2
C HELPER ROUTN, NTBK TOTAL NUMB OF NRBK REALS IN FFT BLOCK
C  (NOTE IRMF=+1 FOR DATA IN COMPLEX STATE, NCNT ELMTS)
C
C BELOW, PROGRAM CAN ACCESS FFT COMPLEX RESULT (AND FORM COMPARISON, IF KNOWN)
      DO 8Ø JB=1,NTBK
      K=(JB-1)*NCNT
      CALL MFREAD(BUFA,NRBK,JB)
C READ 'RECDS' OF NRBK LENGTH, TOTALLING NTBK (RECOMPUTED BY MFPAR)
C  (NOTE, CYBER MFREAD/MFWRIT (USING READMS/WRITMS) CANNOT ACCEPT 'RECDS'
C   OF DIFFERENT LENGTH; OTHERWISE COULD USE NTD1 'RECDS' OF NRD1 HERE)
C
      DO 8Ø I=1,NCNT
      INDEX=J-1
      IF(IPRINT.LE.Ø)GO TO 8Ø
      WRITE(LP,94Ø)INDEX,CBUFA(I)
94Ø   FORMAT(1X,I5,2X,2E13.4)
8Ø    CONTINUE
C
C BELOW, INVERT ISGN AND IDIR FOR INVERSE TRANSFORM (COMPLEX-TO-REAL)
75    ISGN=-ISGN
      IDIR=-IDIR
      SCAL=1./SIZE
      IF(IRMF.NE.Ø)
     X  CALL RMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX, IPAK)
C EITHER ROUTINE RMFFT INVERSE TRANSFORMS ARRAY TO PACKED REAL
C
      IF(IRMF.EQ.Ø)
     X  CALL CMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX)
C OR ROUTINE CMFFT INVERSE TRANSFORMS ARRAY
C
      IF(IPRINT.LE.Ø)GO TO 85
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      WRITE(LP,91Ø)IPRINT,IRMF
      WRITE(LP,92Ø)NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
      WRITE(LP,96Ø)
96Ø   FORMAT(8H   INDEX,4X,8HFFT/2**M,6X,5HINPUT,1ØX,4HDIFF)
C
85    IERR=MFPAR(IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
      NCNT=NRBK
      IF(IRMF.EQ.Ø)NCNT=NCNT/2
C HELPER ROUTN, NTBK TOTAL NUMB OF NRBK REALS IN FFT BLOCK
C   (NCNT IS NUMBER OF ELMTS IN FFT BLOCK, REAL OR CMPLX)
C
C BELOW, COMPARES FFT INVERSE RESULT WITH INITIAL RANDOM INPUT
      DIFM=Ø.
      VALU=Ø.
      DO 9Ø JB=1,NTBK
```

```
      CALL MFREAD(BUFA,NRBK,JB)
C READ 'RECDS' OF NRBK LENGTH, TOTALLING NTBK (RECOMPUTED BY MFPAR)
C  (NOTE, CYBER MFREAD/MFWRIT (USING READMS/WRITMS) CANNOT ACCEPT 'RECDS'
C   OF DIFFERENT LENGTH; OTHERWISE COULD USE NTD1 'RECDS' OF NRD1 HERE)
C
      DO 9Ø I=1,NCNT
      IF(IRMF.NE.Ø)RM=BUFA(I)
      IF(IRMF.EQ.Ø)RM=REAL(CBUFA(I))
      DIF=ABS(RM-VALU)
      IF(DIF.GT.DIFM)DIFM=DIF
      IF(IPRINT.GT.Ø)WRITE(LP,99Ø)I,RM,VALU,DIF
99Ø   FORMAT(1X,I5,3(2X,E13.4))
      VALU=VALU+VINC
9Ø    CONTINUE
C PRINT INVERSE RESULTS (SHOULD BE SAME AS INITIAL RAMP)
C
      IF(IPRINT.GE.Ø)WRITE(LP,95Ø)DIFM,NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
95Ø   FORMAT(1ØH MAX DIFF ,E11.4,1X,
     X  8H  NDIM =,I3,8H  ISGN =,I3,8H  IDIR =,I3,
     X  8H  IBEX =,I3,8H  ICEX =,I3,8H  IPAK =,I3,
     X  1ØH  MEXA() =,3I5)
      IF(DIFM.GT.DIFMG)DIFMG=DIFM
C
      STOP
C
1ØØØ  WRITE(LP,98Ø)IERR
98Ø   FORMAT(25H IBEX FORCED TOO SMALL OR,
     X   3ØH NOT ALL /MFINT/ CORRECT, IERR;I4)
      STOP
      END


      SUBROUTINE MFREAD(BUFA,NB,JB)
C
C  CONTROL DATA 6ØØØ AND CYBER MASS STORE I/O ROUTINES FOR FFT
C
C  (LOGICAL UNIT LUN IN COMMON/FFTCOM/ MUST HAVE BEEN OPENED
C  PREVIOUSLY; EG.      CALL OPENMS(LUN,INDXARRAY,NREC+1,Ø)
C  WHERE NREC=2**(M-IBEX+1) IF IPAK=1 OR LESS IF IPAK=Ø OR -1)
C
C  SEE ALSO ALTERNATIVE SUBROUTINES USING EXTENDED CORE OR LCM
C  (IN ADDITION, GETW, PUTW OR READM, WRITEM MACROS CAN BE USED)
C
C READ BLOCK, INDEX JB, FROM MASS STORE TO BUFA, NB REAL VALUES
C
      COMMON/FFTCOM/LUN,MINDX(512)
      REAL BUFA(NB)
C
      CALL READMS(LUN,BUFA,NB,JB)
      RETURN
C
      ENTRY MFWRIT
C WRITE BLOCK, INDEX JB, FROM BUFA TO MASS STORE, NB REAL VALUES
C
      CALL WRITMS(LUN,BUFA,NB,JB,-1,Ø)
      RETURN
C
      END


      PROGRAM LCMTOM(INPUT,OUTPUT,TAPE6Ø=INPUT,TAPE5=OUTPUT)
C
C  CONTROL DATA 6ØØØ AND CYBER EXTENDED CORE/LCM FFT TEST PROGRAM
C
C  NOTE WELL THAT TYPE COMPLEX DATA MUST EXIST AS ALTERNATING
C  REAL/IMAG/REAL/IMAG... ELEMENTS, BOTH IN MASS STORE AND IN FORTRAN
C  WORKING ARRAY BUFA;  IN THIS FFT, DIFFERENT SUBROUTINES WILL SET
C  DIFFERENT TYPE (REAL OR COMPLEX) FOR ARRAY BUFA.
C
C
      LEVEL 3, LBUFA
      COMMON/FFTCOM/LBUFA(32768)
C
C  LBUFA IN EXTENDED CORE/LCM SIMULATES FAST MASS STORE
```

```
C  DIMENSION LBUFA AS LARGE AS NECESSARY FOR RESULTANT ARRAY
C
      COMMON/MFARG/MEXA(4),NDIM,ISGN,IDIR,SCAL,IBEX,ICEX,IPAK
      COMMON/MFVAL/DIMA(4),TDM1,RDM1,FBLK,TBLK,RBLK,RCOR,SIZE
      COMMON/MFINT/NDMA(4),NTD1,NRD1,NFBK,NTBK,NRBK,NRCR,NSZE
C
C COMMON AREAS /MFARG/,/MFVAL/,/MFINT/ USEFUL FOR RUNNING MASS STORE FFT
C  /MFARG/ HOLDS ARGUMENTS USED IN FFT CALLES, MOSTLY EXPONENTS
C    HELPER ROUTINE MFPAR COMPUTES VALUES FROM /MFARG/ INTO
C  /MFVAL/ (REALS AS SOME LARGE), /MFINT/ (INTEGER EQUIVALENTS IF POSS)
C    OR CAN REVERSE-COMPUTE SOME /MFARG/ FROM /MFVAL/
C       SEE COMMENTS, ROUTINE MFPAR.
C
      COMPLEX CBUFA(4Ø96)
      REAL BUFA(8192)
      EQUIVALENCE (CBUFA(1),BUFA(1))
C
C  BUFA()=CBUFA()    IS WORKING AREA IN CORE STORE FOR FFT AND PROGRAM
C
      LP=5
      IPRINT=Ø
C LP IS PRINTER LOGICAL UNIT, IPRINT=+1 COPIOUS, Ø MAX DIFFERENCES ONLY
C
      IRMF=-1
C IRMF=-1 REAL ROUTINE RMFFT TEST, Ø COMPLEX ROUTINE CMFFT TEST
C
      ISGN=-1
      IDIR=-1
      IPAK=1
C FFT ARGS, ISGN=+/- 1, IDIR=-1 (RMFFT), IDIR=1 OR Ø (CMFFT)
C IPAK=1, Ø OR -1 (RMFFT), NOT USED (CMFFT)
C
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      IF(IPRINT.LE.Ø)WRITE(LP,91Ø)IPRINT,IRMF
91Ø   FORMAT(31H1MASS STORE FFT TEST - IPRINT =,I3,
     X  25H (1=COPIOUS,Ø=MAX DIFFS),,
     X  9H   IRMF =,I3,28H (Ø=CMFFT TEST,1=RMFFT TEST) /)
      DIFMG=Ø.
C
      MEXA(1)=8
      MEXA(2)=6
      NDIM=2
      IBEX=7
      ICEX=13
C MORE FFT ARGS, 2**8 ROWS OF 2**6 ELMTS, 2 DIMEN,
C  FFT MASS STORE BLOCKS 2**IBEX REALS, BUFA 2**ICEX REALS
C
      IERR=MFPAR(IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
      NCNT=NRD1
C HELPER ROUTINE, NCNT NUM OF ELMTS IN FIRST DIMEN, NTD1 TOTAL
C
      M=MFSUM(MEXA,NDIM,99)
      SCAL=1.
      VALU=Ø.
      VINC=1./SIZE
      IF(IRMF.EQ.Ø)GO TO 5Ø
C SWITCH FOR COMPLEX ROUTINE CMFFT AT 5Ø, REAL RMFFT BELOW
C
      DO 4Ø JB=1,NTD1
      DO 35 I=1,NCNT
      BUFA(I)=VALU
35    VALU=VALU+VINC
      CALL MFWRIT(BUFA,NRD1,JB)
4Ø    CONTINUE
C LOAD RAMP FUNCTN IN REAL ARRAY IN 'RECDS' OF NRD1 LENGTH
C  (NOTE, 'LCM' MFREAD/MFWRIT CAN ACCEPT 'RECORDS' OF DIFFERENT LENGTH;
C   SO CAN USE NTD1,NRD1 HERE INSTEAD OF NTBK,NRBK AS IN CYBER MASS STORE)
C
      CALL RMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX, IPAK)
C REAL MASS STORE ROUTINE RMFFT TO TRANSFORM ARRAY TO COMPLEX RESULT
      GO TO 7Ø
C
C BELOW, USE COMPLEX ROUTINE CMFFT,  NCNT NUM OF CMPLX ELMTS IN FFT BLOCK
5Ø    NCNT=NCNT/2
      DO 6Ø JB=1,NTD1
```

```
      DO 55 I=1,NCNT
      CBUFA(I)=CMPLX(VALU,Ø.)
55    VALU=VALU+VINC
      CALL MFWRIT(BUFA,NRD1,JB)
6Ø    CONTINUE
C LOAD RAMP FUNCTN IN COMPLEX ARRAY IN 'RECDS' OF NRD1 LENGTH
C  (NOTE, 'LCM' MFREAD/MFWRIT CAN ACCEPT 'RECORDS' OF DIFFERENT LENGTH;
C   SO CAN USE NTD1,NRD1 HERE INSTEAD OF NTBK,NRBK AS IN CYBER MASS STORE)
C
      CALL CMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX)
C COMPLEX MASS STORE ROUTINE CMFFT TO TRANSFORM ARRAY TO COMPLEX RESULT
C
7Ø    IF(IPRINT.LE.Ø)GO TO 75
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      WRITE(LP,91Ø)IPRINT,IRMF
      WRITE(LP,92Ø)NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
92Ø   FORMAT(8H  NDIM =,I3,8H  ISGN =,I3,8H  IDIR =,I3,
     X  8H  IBEX =,I3,8H  ICEX =,I3,8H  IPAK =,I3,
     X  1ØH  MEXA() =,3I5)
      WRITE(LP,93Ø)
93Ø   FORMAT(8H   INDEX,12X,3HFFT)
C
      IERR=MFPAR(-IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
      NCNT=NRD1/2
C HELPER ROUTINE, NCNT NUM OF ELMTS IN FIRST DIMEN, NTD1 TOTAL
C  (NOTE IRMF=+1 FOR DATA IN COMPLEX STATE, NCNT ELMTS)
C
C BELOW, PROGRAM CAN ACCESS FFT COMPLEX RESULT (AND FORM COMPARISON, IF KNOWN)
      DO 8Ø JB=1,NTD1
      K=(JB-1)*NCNT
      CALL MFREAD(BUFA,NRD1,JB)
C READ 'RECDS' OF NRD1 LENGTH, TOTALLING NTD1 (RECOMPUTED BY MFPAR)
C  (NOTE, 'LCM' MFREAD/MFWRIT CAN ACCEPT 'RECORDS' OF DIFFERENT LENGTH;
C   SO CAN USE NTD1,NRD1 HERE INSTEAD OF NTBK,NRBK AS IN CYBER MASS STORE)
C
      DO 8Ø I=1,NCNT
      INDEX=J-1
      IF(IPRINT.LE.Ø)GO TO 8Ø
      WRITE(LP,94Ø)INDEX,CBUFA(I)
94Ø   FORMAT(1X,I5,2X,2E13.4)
8Ø    CONTINUE
C
C BELOW, INVERT ISGN AND IDIR FOR INVERSE TRANSFORM (COMPLEX-TO-REAL)
75    ISGN=-ISGN
      IDIR=-IDIR
      SCAL=1./SIZE
      IF(IRMF.NE.Ø)
     X  CALL RMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX, IPAK)
C EITHER ROUTINE RMFFT INVERSE TRANSFORMS ARRAY TO PACKED REAL
C
      IF(IRMF.EQ.Ø)
     X  CALL CMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX)
C OR ROUTINE CMFFT INVERSE TRANSFORMS ARRAY
C
      IF(IPRINT.LE.Ø)GO TO 85
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      WRITE(LP,91Ø)IPRINT,IRMF
      WRITE(LP,92Ø)NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
      WRITE(LP,96Ø)
96Ø   FORMAT(8H   INDEX,4X,8HFFT/2**M,6X,5HINPUT,1ØX,4HDIFF)
C
85    IERR=MFPAR(IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
      NCNT=NRD1
      IF(IRMF.EQ.Ø)NCNT=NCNT/2
C HELPER ROUTINE, NCNT NUM OF ELMTS IN FIRST DIMEN, NTD1 TOTAL
C   (NCNT IS NUMBER OF ELMTS IN FFT BLOCK, REAL OR CMPLX)
C
C BELOW, COMPARES FFT INVERSE RESULT WITH INITIAL RANDOM INPUT
      DIFM=Ø.
      VALU=Ø.
      DO 9Ø JB=1,NTD1
      CALL MFREAD(BUFA,NRD1,JB)
```

```
C READ 'RECDS' OF NRD1 LENGTH, TOTALLING NTD1 (RECOMPUTED BY MFPAR)
C  (NOTE, 'LCM' MFREAD/MFWRIT CAN ACCEPT 'RECORDS' OF DIFFERENT LENGTH;
C   SO CAN USE NTD1,NRD1 HERE INSTEAD OF NTBK,NRBK AS IN CYBER MASS STORE)
C
      DO 9Ø I=1,NCNT
      IF(IRMF.NE.Ø)RM=BUFA(I)
      IF(IRMF.EQ.Ø)RM=REAL(CBUFA(I))
      DIF=ABS(RM-VALU)
      IF(DIF.GT.DIFM)DIFM=DIF
      IF(IPRINT.GT.Ø)WRITE(LP,99Ø)I,RM,VALU,DIF
99Ø   FORMAT(1X,I5,3(2X,E13.4))
      VALU=VALU+VINC
9Ø    CONTINUE
C PRINT INVERSE RESULTS (SHOULD BE SAME AS INITIAL RAMP)
C
      IF(IPRINT.GE.Ø)WRITE(LP,95Ø)DIFM,NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
95Ø   FORMAT(1ØH MAX DIFF ,E11.4,1X,
     X  8H  NDIM =,I3,8H  ISGN =,I3,8H  IDIR =,I3,
     X  8H  IBEX =,I3,8H  ICEX =,I3,8H  IPAK =,I3,
     X  1ØH  MEXA() =,3I5)
      IF(DIFM.GT.DIFMG)DIFMG=DIFM
C
      STOP
C
1ØØØ  WRITE(LP,98Ø)IERR
98Ø   FORMAT(25H IBEX FORCED TOO SMALL OR,
     X   3ØH NOT ALL /MFINT/ CORRECT, IERR,I4)
      STOP
      END


      SUBROUTINE MFREAD(BUFA,NB,JB)
C
C  CONTROL DATA 6ØØØ AND CYBER EXTENDED CORE STORE I/O ROUTINES FOR FFT
C
C  (EXTENDED CORE OR LCM TAKES PLACE OF MASS STORE -
C   DIMENSION LBUFA AS LARGE AS NECESSARY FOR LARGE ARRAYS)
C
C  ALTERNATIVELY, EXTENDED CORE USE COULD BE COMBINED WITH
C  READM/WRITEM MACROS TO ACCESS LARGER FILE WHEN NECESSARY
C  (USE VIRTUAL MEMORY ALGORITHM - MANY SECTORS HELD IN
C    EXTENDED CORE AND ONLY ACCESSED FROM DISC IF NOT PRESENT).
C
C READ BLOCK, INDEX JB, FROM EXTENDED CORE TO BUFA, NB REAL VALUES
C
      LEVEL 3, LBUFA
      COMMON/FFTCOM/LBUFA(32768)
      REAL BUFA(NB)
C
      CALL MOVLEV(LBUFA((JB-1)*NB+1),BUFA,NB)
      RETURN
C
      ENTRY MFWRIT
C WRITE BLOCK, INDEX JB, FROM BUFA TO EXTENDED CORE, NB REAL VALUES
C
      CALL MOVLEV(BUFA,LBUFA((JB-1)*NB+1),NB)
      RETURN
C
      END


C
C PDP 11 MASS STORE I/O FFT SAMPLE TEST PROGRAM
C
C  NOTE WELL THAT TYPE COMPLEX DATA MUST EXIST AS ALTERNATING
C  REAL/IMAG/REAL/IMAG... ELEMENTS, BOTH IN MASS STORE AND IN FORTRAN
C  WORKING ARRAY BUFA;  IN THIS FFT, DIFFERENT SUBROUTINES WILL SET
C  DIFFERENT TYPE (REAL OR COMPLEX) FOR ARRAY BUFA.
C
C
      COMMON/FFTCOM/LUN
C
C COMMON /FFTCOM/ HOLDS LOGICAL UNIT NUMBER FOR MASS STORE I/O
C
```

```
      COMMON/MFARG/MEXA(4),NDIM,ISGN,IDIR,SCAL,IBEX,ICEX,IPAK
      COMMON/MFVAL/DIMA(4),TDM1,RDM1,FBLK,TBLK,RBLK,RCOR,SIZE
      COMMON/MFINT/NDMA(4),NTD1,NRD1,NFBK,NTBK,NRBK,NRCR,NSZE
C
C COMMON AREAS /MFARG/,/MFVAL/,/MFINT/ USEFUL FOR RUNNING MASS STORE FFT
C  /MFARG/ HOLDS ARGUMENTS USED IN FFT CALLES, MOSTLY EXPONENTS
C    HELPER ROUTINE MFPAR COMPUTES VALUES FROM /MFARG/ INTO
C  /MFVAL/ (REALS AS SOME LARGE), /MFINT/ (INTEGER EQUIVALENTS IF POSS)
C    OR CAN REVERSE-COMPUTE SOME /MFARG/ FROM /MFVAL/
C       SEE COMMENTS, ROUTINE MFPAR.
C
      COMPLEX CBUFA(1Ø24)
      REAL BUFA(2Ø48)
      EQUIVALENCE (CBUFA(1),BUFA(1))
C
C  BUFA()=CBUFA()    IS WORKING AREA IN CORE STORE FOR FFT AND PROGRAM
C
      LUN=1
C LUN IS LOGICAL UNIT FOR PDP 11 MASS STORE I/O
C
      LP=5
      IPRINT=Ø
C LP IS PRINTER LOGICAL UNIT, IPRINT=+1 COPIOUS, Ø MAX DIFFERENCES ONLY
C
      IRMF=-1
C IRMF=-1 REAL ROUTINE RMFFT TEST, Ø COMPLEX ROUTINE CMFFT TEST
C
      ISGN=-1
      IDIR=-1
      IPAK=1
C FFT ARGS, ISGN=+/- 1, IDIR=-1 (RMFFT), IDIR=1 OR Ø (CMFFT)
C IPAK=1, Ø OR -1 (RMFFT), NOT USED (CMFFT)
C
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      IF(IPRINT.LE.Ø)WRITE(LP,91Ø)IPRINT,IRMF
91Ø   FORMAT(31H1MASS STORE FFT TEST - IPRINT =,I3,
     X  25H (1=COPIOUS,Ø=MAX DIFFS),,
     X  9H   IRMF =,I3,28H (Ø=CMFFT TEST,1=RMFFT TEST) /)
      DIFMG=Ø.
C
      MEXA(1)=8
      MEXA(2)=6
      NDIM=2
      IBEX=7
      ICEX=11
C MORE FFT ARGS, 2**8 ROWS OF 2**6 ELMTS, 2 DIMEN,
C  FFT MASS STORE BLOCKS 2**IBEX REALS, BUFA 2**ICEX REALS
C
      IERR=MFPAR(IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
      CALL ASSIGN(LUN,'FFTEST.DAT',Ø)
      NWD=NRBK*2
      DEFINE FILE LUN(NFBK,NWD,U,INDX)
C PDP 11 MASS STORE OPENED WITH NUM REC=NFBK, NRBK*2 WORDS PER REC
C  (NOTE HELPER ROUTN MFPAR RETURNS NFBK AS MAXIMUM FILE SIZE)
C
      NCNT=NRBK
C HELPER ROUTINE, NCNT NUM OF ELMTS IN FFT BLOCK, NTBK TOTAL BLOCKS
C
      M=MFSUM(MEXA,NDIM,99)
      SCAL=1.
      VALU=Ø.
      VINC=1./SIZE
      IF(IRMF.EQ.Ø)GO TO 5Ø
C SWITCH FOR COMPLEX ROUTINE CMFFT AT 5Ø, REAL RMFFT BELOW
C
      DO 4Ø JB=1,NTBK
      DO 35 I=1,NCNT
      BUFA(I)=VALU
35    VALU=VALU+VINC
      CALL MFWRIT(BUFA,NRBK,JB)
4Ø    CONTINUE
C LOAD RAMP FUNCTN IN REAL ARRAY IN 'RECDS' OF NRBK LENGTH
C  (NOTE, PDP 11 MFREAD/MFWRIT (USING FORTRAN I/O) CANNOT ACCEPT 'RECDS'
C   OF DIFFERENT LENGTH; OTHERWISE COULD USE NTD1 'RECDS' OF NRD1 HERE)
C
```

```
      CALL RMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX, IPAK)
C REAL MASS STORE ROUTINE RMFFT TO TRANSFORM ARRAY TO COMPLEX RESULT
      GO TO 7Ø
C
C BELOW, USE COMPLEX ROUTINE CMFFT,  NCNT NUM OF CMPLX ELMTS IN FFT BLOCK
5Ø    NCNT=NCNT/2
      DO 6Ø JB=1,NTBK
      DO 55 I=1,NCNT
      CBUFA(I)=CMPLX(VALU,Ø.)
55    VALU=VALU+VINC
      CALL MFWRIT(BUFA,NRBK,JB)
6Ø    CONTINUE
C LOAD RAMP FUNCTN IN COMPLEX ARRAY IN 'RECDS' OF NRBK LENGTH
C  (NOTE, PDP 11 MFREAD/MFWRIT (USING FORTRAN I/O) CANNOT ACCEPT 'RECDS'
C   OF DIFFERENT LENGTH; OTHERWISE COULD USE NTD1 'RECDS' OF NRD1 HERE)
C
      CALL CMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX)
C COMPLEX MASS STORE ROUTINE CMFFT TO TRANSFORM ARRAY TO COMPLEX RESULT
C
7Ø    IF(IPRINT.LE.Ø)GO TO 75
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      WRITE(LP,91Ø)IPRINT,IRMF
      WRITE(LP,92Ø)NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
92Ø   FORMAT(8H  NDIM =,I3,8H  ISGN =,I3,8H  IDIR =,I3,
     X  8H  IBEX =,I3,8H  ICEX =,I3,8H  IPAK =,I3,
     X  1ØH  MEXA() =,3I5)
      WRITE(LP,93Ø)
93Ø   FORMAT(8H   INDEX,12X,3HFFT)
C
      IERR=MFPAR(-IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
      IF(IERR.NE.Ø)GO TO 1ØØØ
      NCNT=NRBK/2
C HELPER ROUTN, NTBK TOTAL NUMB OF NRBK REALS IN FFT BLOCK
C  (NOTE IRMF=+1 FOR DATA IN COMPLEX STATE, NCNT ELMTS)
C
C BELOW, PROGRAM CAN ACCESS FFT COMPLEX RESULT (AND FORM COMPARISON, IF KNOWN)
      DO 8Ø JB=1,NTBK
      K=(JB-1)*NCNT
      CALL MFREAD(BUFA,NRBK,JB)
C READ 'RECDS' OF NRBK LENGTH, TOTALLING NTBK (RECOMPUTED BY MFPAR)
C  (NOTE, PDP 11 MFREAD/MFWRIT (USING FORTRAN I/O) CANNOT ACCEPT 'RECDS'
C   OF DIFFERENT LENGTH; OTHERWISE COULD USE NTD1 'RECDS' OF NRD1 HERE)
C
      DO 8Ø I=1,NCNT
      INDEX=J-1
      IF(IPRINT.LE.Ø)GO TO 8Ø
      WRITE(LP,94Ø)INDEX,CBUFA(I)
94Ø   FORMAT(1X,I5,2X,2E13.4)
8Ø    CONTINUE
C
C BELOW, INVERT ISGN AND IDIR FOR INVERSE TRANSFORM (COMPLEX-TO-REAL)
75    ISGN=-ISGN
      IDIR=-IDIR
      SCAL=1./SIZE
      IF(IRMF.NE.Ø)
     X  CALL RMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX, IPAK)
C EITHER ROUTINE RMFFT INVERSE TRANSFORMS ARRAY TO PACKED REAL
C
      IF(IRMF.EQ.Ø)
     X  CALL CMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX)
C OR ROUTINE CMFFT INVERSE TRANSFORMS ARRAY
C
      IF(IPRINT.LE.Ø)GO TO 85
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      WRITE(LP,91Ø)IPRINT,IRMF
      WRITE(LP,92Ø)NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
      WRITE(LP,96Ø)
96Ø   FORMAT(8H   INDEX,4X,8HFFT/2**M,6X,5HINPUT,1ØX,4HDIFF)
C
85    IERR=MFPAR(IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
      NCNT=NRBK
      IF(IRMF.EQ.Ø)NCNT=NCNT/2
```

```
C HELPER ROUTN, NTBK TOTAL NUMB OF NRBK REALS IN FFT BLOCK
C   (NCNT IS NUMBER OF ELMTS IN FFT BLOCK, REAL OR CMPLX)
C
C BELOW, COMPARES FFT INVERSE RESULT WITH INITIAL RANDOM INPUT
      DIFM=Ø.
      VALU=Ø.
      DO 9Ø JB=1,NTBK
      CALL MFREAD(BUFA,NRBK,JB)
C READ 'RECDS' OF NRBK LENGTH, TOTALLING NTBK (RECOMPUTED BY MFPAR)
C  (NOTE, PDP 11 MFREAD/MFWRIT (USING FORTRAN I/O) CANNOT ACCEPT 'RECDS'
C   OF DIFFERENT LENGTH; OTHERWISE COULD USE NTD1 'RECDS' OF NRD1 HERE)
C
      DO 9Ø I=1,NCNT
      IF(IRMF.NE.Ø)RM=BUFA(I)
      IF(IRMF.EQ.Ø)RM=REAL(CBUFA(I))
      DIF=ABS(RM-VALU)
      IF(DIF.GT.DIFM)DIFM=DIF
      IF(IPRINT.GT.Ø)WRITE(LP,99Ø)I,RM,VALU,DIF
99Ø   FORMAT(1X,I5,3(2X,E13.4))
      VALU=VALU+VINC
9Ø    CONTINUE
C PRINT INVERSE RESULTS (SHOULD BE SAME AS INITIAL RAMP)
C
      IF(IPRINT.GE.Ø)WRITE(LP,95Ø)DIFM,NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
95Ø   FORMAT(1ØH MAX DIFF ,E11.4,1X,
     X  8H  NDIM =,I3,8H  ISGN =,I3,8H  IDIR =,I3,
     X  8H  IBEX =,I3,8H  ICEX =,I3,8H  IPAK =,I3,
     X  1ØH  MEXA() =,3I5)
      IF(DIFM.GT.DIFMG)DIFMG=DIFM
C
      STOP
C
1ØØØ  WRITE(LP,98Ø)IERR
98Ø   FORMAT(25H IBEX FORCED TOO SMALL OR,
     X   3ØH NOT ALL /MFINT/ CORRECT, IERR,I4)
      STOP
      END


      SUBROUTINE MFREAD(BUFA,NB,JB)
C
C  PDP 11 FORTRAN DIRECT ACCESS READ ROUTINE FOR FFT
C
C  (LOGICAL UNIT LUN IN COMMON/FFTCOM/ MUST HAVE BEEN OPENED
C  PREVIOUSLY; EG.      CALL ASSIGN(LUN,'FILENAME',Ø)
C  AND                  DEFINE FILE LUN(NREC,NWD,U,INDX)
C  WHERE NWD=2**(IBEX+1) AND NREC=2**(M-IBEX+1) OR LESS IF IPAK=Ø OR -1)
C
C  SEE ALSO ALTERNATIVE, FAST MACRO I/O SUBROUTINES
C  (THESE ARE TO BE PREFERRED, SINCE SPEED 2 TO 1Ø TIMES BETTER)
C
C READ BLOCK, INDEX JB, FROM MASS STORE TO BUFA, NB REAL VALUES
C
      COMMON/FFTCOM/LUN
      REAL BUFA(NB)
C
      READ(LUN'JB)BUFA
      RETURN
C
      END


      SUBROUTINE MFWRIT(BUFA,NB,JB)
C
C  PDP 11 FORTRAN DIRECT ACCESS WRITE ROUTINE FOR FFT
C
C  (LOGICAL UNIT LUN IN COMMON/FFTCOM/ MUST HAVE BEEN OPENED
C  PREVIOUSLY; EG.      CALL ASSIGN(LUN,'FILENAME',Ø)
C  AND                  DEFINE FILE LUN(NREC,NWD,U,INDX)
C  WHERE NWD=2**(IBEX+1) AND NREC=2**(M-IBEX+1) OR LESS IF IPAK=Ø OR -1)
C
C  SEE ALSO ALTERNATIVE, FAST MACRO I/O SUBROUTINES
C  (THESE ARE TO BE PREFERRED, SINCE SPEED 2 TO 1Ø TIMES BETTER)
C
```

```
C WRITE BLOCK, INDEX JB, FROM BUFA TO MASS STORE, NB REAL VALUES
C
      COMMON/FFTCOM/LUN
      REAL BUFA(NB)
C
      WRITE(LUN'JB)BUFA
      RETURN
C
      END


C
C PDP 11 FAST MACRO I/O FFT SAMPLE TEST PROGRAM
C (REQUIRES CALL TO SYSTEM MACRO TO OPEN FILE FOR I/O - SEE LINE 7Ø)
C
C  NOTE THAT FAST MACRO I/O IS MORE EFFEICIENT THAN STANDARD FORTRAN I/O
C
C  NOTE WELL THAT TYPE COMPLEX DATA MUST EXIST AS ALTERNATING
C  REAL/IMAG/REAL/IMAG... ELEMENTS, BOTH IN MASS STORE AND IN FORTRAN
C  WORKING ARRAY BUFA;  IN THIS FFT, DIFFERENT SUBROUTINES WILL SET
C  DIFFERENT TYPE (REAL OR COMPLEX) FOR ARRAY BUFA.
C
C
      COMMON/FFTCOM/IFDB,IOERR
C
C COMMON /FFTCOM/ HOLDS FDB ADDRESS FOR MACRO I/O (RSX11M), CHANNEL (RT11)
C     (SEE COMMENTS IN FAST MACRO ROUTINE MFREAD/MFWRIT)
C
      COMMON/MFARG/MEXA(4),NDIM,ISGN,IDIR,SCAL,IBEX,ICEX,IPAK
      COMMON/MFVAL/DIMA(4),TDM1,RDM1,FBLK,TBLK,RBLK,RCOR,SIZE
      COMMON/MFINT/NDMA(4),NTD1,NRD1,NFBK,NTBK,NRBK,NRCR,NSZE
C
C COMMON AREAS /MFARG/,/MFVAL/,/MFINT/ USEFUL FOR RUNNING MASS STORE FFT
C  /MFARG/ HOLDS ARGUMENTS USED IN FFT CALLES, MOSTLY EXPONENTS
C    HELPER ROUTINE MFPAR COMPUTES VALUES FROM /MFARG/ INTO
C  /MFVAL/ (REALS AS SOME LARGE), /MFINT/ (INTEGER EQUIVALENTS IF POSS)
C    OR CAN REVERSE-COMPUTE SOME /MFARG/ FROM /MFVAL/
C       SEE COMMENTS, ROUTINE MFPAR.
C
      COMPLEX CBUFA(1Ø24)
      REAL BUFA(2Ø48)
      EQUIVALENCE (CBUFA(1),BUFA(1))
C
C  BUFA()=CBUFA()    IS WORKING AREA IN CORE STORE FOR FFT AND PROGRAM
C
      LUN=1
C LUN IS LOGICAL UNIT FOR PDP 11 MASS STORE I/O
C
      LP=5
      IPRINT=Ø
C LP IS PRINTER LOGICAL UNIT, IPRINT=+1 COPIOUS, Ø MAX DIFFERENCES ONLY
C
      IRMF=-1
C IRMF=-1 REAL ROUTINE RMFFT TEST, Ø COMPLEX ROUTINE CMFFT TEST
C
      ISGN=-1
      IDIR=-1
      IPAK=1
C FFT ARGS, ISGN=+/- 1, IDIR=-1 (RMFFT), IDIR=1 OR Ø (CMFFT)
C IPAK=1, Ø OR -1 (RMFFT), NOT USED (CMFFT)
C
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      IF(IPRINT.LE.Ø)WRITE(LP,91Ø)IPRINT,IRMF
91Ø   FORMAT(31H1MASS STORE FFT TEST - IPRINT =,I3,
     X  25H (1=COPIOUS,Ø=MAX DIFFS),,
     X  9H   IRMF =,I3,28H (Ø=CMFFT TEST,1=RMFFT TEST) /)
      DIFMG=Ø.
C
      MEXA(1)=8
      MEXA(2)=6
      NDIM=2
      IBEX=7
      ICEX=11
C MORE FFT ARGS, 2**8 ROWS OF 2**6 ELMTS, 2 DIMEN,
C  FFT MASS STORE BLOCKS 2**IBEX REALS, BUFA 2**ICEX REALS
C
```

```
      IERR=MFPAR(IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
C   (NOTE HELPER ROUTN MFPAR RETURNS NFBK AS MAXIMUM FILE SIZE)
C
      STOP 'ASSIGN AND OPEN FILE HERE'
C DELETE ABOVE LINE AND INVOKE SYSTEM MACRO 'OPEN$' (RSX) OR '.OPEN (RT11)
C
C FOR EXAMPLE, A FORTRAN CALLABLE SUBROUTINE 'MFOPEN' COULD OPEN FILE
C  'FFTTEST.DAT', RETURNING IFDB=FDB ADDRESS (RSX11M) IN COMMON/FFTCOM/
C  OF CHANNEL NUMBER (RT11), FOR USE BY MACRO MFREAD/MFWRIT, THUS:
C
C     CALL MFOPEN(LUN,'FFTTEST.DAT',IFDB)
C
      IOERR=Ø
C PRESET IOERR IN /FFTCOM/; ZERO IF NO ERRORS IN I/O
C
      NCNT=NRD1
C HELPER ROUTINE, NCNT NUM OF ELMTS IN 'FIRST' DIMEN, NTD1 TOTAL BLOCKS
C
      M=MFSUM(MEXA,NDIM,99)
      SCAL=1.
      VALU=Ø.
      VINC=1./SIZE
      IF(IRMF.EQ.Ø)GO TO 5Ø
C SWITCH FOR COMPLEX ROUTINE CMFFT AT 5Ø, REAL RMFFT BELOW
C
      DO 4Ø JB=1,NTD1
      DO 35 I=1,NCNT
      BUFA(I)=VALU
35    VALU=VALU+VINC
      CALL MFWRIT(BUFA,NRD1,JB)
4Ø    CONTINUE
C LOAD RAMP FUNCTN IN REAL ARRAY IN 'RECDS' OF NRD1 LENGTH
C  (NOTE, 'MACRO' MFREAD/MFWRIT CAN ACCEPT 'RECORDS' OF DIFFERENT LENGTH;
C   SO CAN USE NTD1,NRD1 HERE INSTEAD OF NTBK,NRBK AS IN FORTRAN MASS STORE)
C
      CALL RMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX, IPAK)
C REAL MASS STORE ROUTINE RMFFT TO TRANSFORM ARRAY TO COMPLEX RESULT
      GO TO 7Ø
C
C BELOW, USE COMPLEX ROUTINE CMFFT,  NCNT NUM OF CMPLX ELMTS IN 'FIRST' DIMEN
5Ø    NCNT=NCNT/2
      DO 6Ø JB=1,NTD1
      DO 55 I=1,NCNT
      CBUFA(I)=CMPLX(VALU,Ø.)
55    VALU=VALU+VINC
      CALL MFWRIT(BUFA,NRD1,JB)
6Ø    CONTINUE
C LOAD RAMP FUNCTN IN COMPLEX ARRAY IN 'RECDS' OF NRD1 LENGTH
C  (NOTE, 'MACRO' MFREAD/MFWRIT CAN ACCEPT 'RECORDS' OF DIFFERENT LENGTH;
C   SO CAN USE NTD1,NRD1 HERE INSTEAD OF NTBK,NRBK AS IN FORTRAN MASS STORE)
C
      CALL CMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX)
C COMPLEX MASS STORE ROUTINE CMFFT TO TRANSFORM ARRAY TO COMPLEX RESULT
C
7Ø    IF(IPRINT.LE.Ø)GO TO 75
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      WRITE(LP,91Ø)IPRINT,IRMF
      WRITE(LP,92Ø)NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
92Ø   FORMAT(8H  NDIM =,I3,8H  ISGN =,I3,8H  IDIR =,I3,
     X  8H  IBEX =,I3,8H  ICEX =,I3,8H  IPAK =,I3,
     X  1ØH  MEXA() =,3I5)
      WRITE(LP,93Ø)
93Ø   FORMAT(8H   INDEX,12X,3HFFT)
C
      IERR=MFPAR(-IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
      NCNT=NRD1/2
C HELPER ROUTN, NTD1 TOTAL NUMB OF NRD1 REALS IN 'FIRST' DIMEN
C  (NOTE IRMF=+1 FOR DATA IN COMPLEX STATE, NCNT ELMTS)
C
C BELOW, PROGRAM CAN ACCESS FFT COMPLEX RESULT (AND FORM COMPARISON, IF KNOWN)
      DO 8Ø JB=1,NTD1
      K=(JB-1)*NCNT
      CALL MFREAD(BUFA,NRD1,JB)
```

```
C READ 'RECDS' OF NRD1 LENGTH, TOTALLING NTD1 (RECOMPUTED BY MFPAR)
C  (NOTE, 'MACRO' MFREAD/MFWRIT CAN ACCEPT 'RECORDS' OF DIFFERENT LENGTH;
C   SO CAN USE NTD1,NRD1 HERE INSTEAD OF NTBK,NRBK AS IN FORTRAN MASS STORE)
C
      DO 8Ø I=1,NCNT
      INDEX=J-1
      IF(IPRINT.LE.Ø)GO TO 8Ø
      WRITE(LP,94Ø)INDEX,CBUFA(I)
94Ø   FORMAT(1X,I5,2X,2E13.4)
8Ø    CONTINUE
C
C BELOW, INVERT ISGN AND IDIR FOR INVERSE TRANSFORM (COMPLEX-TO-REAL)
75    ISGN=-ISGN
      IDIR=-IDIR
      SCAL=1./SIZE
      IF(IRMF.NE.Ø)
     X  CALL RMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX, IPAK)
C EITHER ROUTINE RMFFT INVERSE TRANSFORMS ARRAY TO PACKED REAL
C
      IF(IRMF.EQ.Ø)
     X  CALL CMFFT (MEXA,NDIM,ISGN,IDIR,SCAL, BUFA,IBEX,ICEX)
C OR ROUTINE CMFFT INVERSE TRANSFORMS ARRAY
C
      IF(IPRINT.LE.Ø)GO TO 85
C BELOW, PRINT HEADINGS FOR TEST OUTPUT
      WRITE(LP,91Ø)IPRINT,IRMF
      WRITE(LP,92Ø)NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
      WRITE(LP,96Ø)
96Ø   FORMAT(8H    INDEX,4X,8HFFT/2**M,6X,5HINPUT,1ØX,,4HDIFF)
C
85    IERR=MFPAR(IRMF,Ø)
      IF(IERR.NE.Ø)GO TO 1ØØØ
      NCNT=NRD1
      IF(IRMF.EQ.Ø)NCNT=NCNT/2
C HELPER ROUTN, NTD1 TOTAL NUMB OF NRD1 REALS IN 'FIRST' DIMEN
C   (NCNT IS NUMBER OF ELMTS IN 'FIRST' DIMEN, REAL OR CMPLX)
C
C BELOW, COMPARES FFT INVERSE RESULT WITH INITIAL RANDOM INPUT
      DIFM=Ø.
      VALU=Ø.
      DO 9Ø JB=1,NTD1
      CALL MFREAD(BUFA,NRD1,JB)
C READ 'RECDS' OF NRD1 LENGTH, TOTALLING NTD1 (RECOMPUTED BY MFPAR)
C  (NOTE, 'MACRO' MFREAD/MFWRIT CAN ACCEPT 'RECORDS' OF DIFFERENT LENGTH;
C   SO CAN USE NTD1,NRD1 HERE INSTEAD OF NTBK,NRBK AS IN FORTRAN MASS STORE)
C
      DO 9Ø I=1,NCNT
      IF(IRMF.NE.Ø)RM=BUFA(I)
      IF(IRMF.EQ.Ø)RM=REAL(CBUFA(I))
      DIF=ABS(RM-VALU)
      IF(DIF.GT.DIFM)DIFM=DIF
      IF(IPRINT.GT.Ø)WRITE(LP,99Ø)I,RM,VALU,DIF
99Ø   FORMAT(1X,I5,3(2X,E13.4))
      VALU=VALU+VINC
9Ø    CONTINUE
C PRINT INVERSE RESULTS (SHOULD BE SAME AS INITIAL RAMP)
C
      IF(IPRINT.GE.Ø)WRITE(LP,95Ø)DIFM,NDIM,ISGN,IDIR,IBEX,ICEX,IPAK,
     X  (MEXA(J),J=1,NDIM)
95Ø   FORMAT(1ØH MAX DIFF ,E11.4,1X,
     X  8H  NDIM =,I3,8H  ISGN =,I3,8H  IDIR =,I3,
     X  8H  IBEX =,I3,8H  ICEX =,I3,8H  IPAK =,I3,
     X  1ØH  MEXA() =,3I5)
      IF(DIFM.GT.DIFMG)DIFMG=DIFM
C
      STOP
C
1ØØØ  WRITE(LP,98Ø)IERR
98Ø   FORMAT(25H IBEX FORCED TOO SMALL OR,
     X   3ØH NOT ALL /MFINT/ CORRECT, IERR,I4)
      STOP
      END
```

```
;
;  PDP11 RSX11M OR RT11 FAST MACRO I/O ROUTINES FOR FFT
;
;     SUBROUTINE MFREAD(BUFA,NB,JB)
;     SUBROUTINE MFWRIT(BUFA,NB,JB)
;
;C READ  BLOCK, INDEX JB, FROM MASS STORE TO BUFA, NB REAL VALUES
;C WRITE BLOCK, INDEX JB, FROM BUFA TO MASS STORE, NB REAL VALUES
;
;     COMMON/FFTCOM/IFDB,IOERR  (RSX11 FDB ADDRESS, ERROR)
;OR   COMMON/FFTCOM/ICHAN,IOERR (RT11  CHANNEL NUM, ERROR)
;
;NOTE1: FILE MUST BE PROPERLY OPEN FOR READ/WRITE ACCESS AND WITH
;       RSX11M FDB ADDRESS OR RT11 CHANNEL NUMBER IN FIRST WORD
;       OF COMMON/FFTCOM/ (IE. INVOKE RSX11M OR RT11 OPEN MACROS).
;
;NOTE2: I/O ERRORS, IF THEY OCCUR, ARE RETURNED IN SECOND WORD
;       OF COMMON/FFTCOM/.  THIS WORD (IOERR) REMAINS UNCHANGED
;       IF NO ERROR OCCURS - THUS, IF PRESET TO Ø BEFORE CALLING
;       MFREAD/MFWRIT (OR MASS STORE FFT), WILL FINALLY BE
;       Ø IF NO ERRORS, OR NON ZERO=LAST ERROR (SEE FOLLOWING LABEL L4:).
;
;NOTE3: NB MUST BE AN INTEGRAL POWER OF 2; 128 OR MORE MOST EFFICIENT
;       (NB.LE.64 USES LOCAL 256 WORD JBUF TO HOLD SECTOR;
;        WRITE ALWAYS WRITES THIS SECTOR TO FILE; SLOW BUT SAFE).
;
;NOTE4: READY TO ASSEMBLE FOR RSX11M. FOR RT11, ERASE FOLLOWING LINE:
       RSX=Ø
;
       .IF     DF,RSX
; CONDITIONAL SECTION FOR RSX11M (V3)
       .TITLE  MFREAD(RSX11M)
       .MCALL  READ$,WRITE$,WAIT$
       .PSECT  FFTCOM,RW,D,GBL,REL,OVR
IFDB:  .WORD   Ø        ;FDB ADDRESS
IOERR: .WORD   Ø        ;ERROR FLAG (UNCHANGED IF NO ERROR)
       .PSECT
       .ENDC
;
       .IF     NDF,RSX
; CONDITIONAL SECTION FOR RT11 (V3)
       .TITLE  MFREAD(RT11)
       .MCALL  .READW,.WRITW
       ERRBYT=52
       .PSECT  FFTCOM,RW,D,GBL,REL,OVR
ICHAN: .WORD   Ø        ;CHANNEL NUMBER
IOERR: .WORD   Ø        ;ERROR FLAG (UNCHANGED IF NO ERROR)
       .PSECT  MFREAD,RW,I,LCL,REL,OVR
       .ENDC
;
       .GLOBL  MFREAD,MFWRIT
;
MFREAD:  MOV     #1,RØ           ;MFREAD SETS JSW=1
       BR      L1
;
MFWRIT:  MOV     #-1,RØ          ;MFWRIT SETS JSW=-1
;
L1:    MOV     RØ,JSW          ;HOLD JSW
       TST     (R5)+           ;IGNORE ARGUMENT COUNT
       MOV     (R5)+,R3        ;R3=ADDRESS OF 'BUFA'
       MOV     @(R5)+,R2       ;R2='NB'
       MOV     @(R5)+,R1       ;R1='JB'
; FINISH ACCESSING SUBROUTINE ARGUMENTS
;
       DEC     R1              ;R1=JB-1
       ASL     R2              ;R2=WORD COUNT, NB*2
       MOV     R2,R5           ;COPY TO R5
       BEQ     L6              ;FINISH IF COUNT ZERO
       CLR     RØ              ;CLEAR RØ FOR WORD OFFSET
       CMP     R5,#256.        ;CHECK WHETHER WHOLE SECTORS
       BLT     L12             ;GO TO L12 IF NOT (MORE COMPLEX)
;
; BELOW, SCALE R1 TO SECTOR OFFSET (ASSUMES NWD POWER OF 2)
L1Ø:   BIT     #256.,R5        ;LOOK FOR BIT AT 256.
       BNE     L11             ;R1 COMPLETE WHEN FOUND
       ASL     R1              ;SCALE R1 ACCORDING TO R5
```

```
        ASR     R5
        BR      L1Ø
;
; NOW HAVE      R1=SECTOR START (FIRST=Ø)
;               R2=WORD COUNT
;               R3=BUFA ADDRESS
;
L11:    MOV     JSW,JSWIO       ;SET UP FOR READ/WRITE
        BGT     L13
        MOV     #-1,SECHLD      ;DIRECT WRITE, ENSURE JBUF LOOKS EMPTY
L13:    JSR     PC,INOUT        ;CALL DIRECT INPUT/OUTPUT
L6:     RTS     PC              ;RETURN FROM SUBROUTINE
; END OF SIMPLE DIRECT INPUT/OUTPUT
;
        .IF     DF,RSX
; CONDITIONAL SECTION FOR RSX11M
INOUT:  ASL     R2              ;RSX11M NEEDS BYTE COUNT=NB*4
        INC     R1              ;RSX11M SECTOR STARTS WITH 1
        MOV     R1,SECTL        ;HOLD AS LOWER SECTOR VALUE
        MOV     IFDB,R4         ;R4=FDB ADDRESS
        MOVB    F.LUN(R4),R1    ;USE LUN FOR EVENT FLAG
;
        TST     JSWIO
        BLT     L5              ;BRANCH IF WRITE
;
        MOV     F.EFBK+2(R4),RØ ;GET LOW SECTOR OF EOF
        DEC     RØ              ;ALLOW FOR HEADER SECTOR
        CMP     RØ,SECTL        ;TEST IF SECTOR EXISTS YET
        BLT     L7              ;NO READ IF DOES NOT EXIST
;
        READ$   R4,R3,R2,#SECT,R1 ;FDB,BUF,BYTCNT,SECT,EVFLG
;
L4:     MOV     IFDB,R4         ;R4=FDB ADDRESS
        MOVB    F.LUN(R4),R1    ;USE LUN FOR EVENT FLAG
        WAIT$   R4,R1           ;WAIT FOR I/O FINISH
        BCC     L7              ;NO ERROR IF CARRY CLEAR
;
        MOV     IFDB,R4         ;R4=FDB ADDRESS
        MOVB    F.ERR(R4),R4    ;HOLD NEGATIVE ERROR CODE
        MOV     R4,IOERR        ;IN IOERR IN COMMON/FFTCOM/
L7:     RTS     PC
;
L5:     WRITE$  R4,R3,R2,#SECT,R1 ;FDB,BUF,BYTCNT,SECT,EVFLG
        BR      L4
;
SECT:   .WORD   Ø       ;HOLDS SECTOR VALUE (HIGHER, ALWAYS Ø)
SECTL:  .WORD   Ø       ;(LOWER, COMPUTED)
        .ENDC
;
        .IF     NDF,RSX
;CONDITIONAL SECTION FOR RT11
INOUT:  MOV     ICHAN,R4        ;R4=CHANNEL NUMBER
;
        TST     JSWIO
        BLT     L5              ;BRANCH IF WRITE
;
        .READW  #AREA,R4,R3,R2,R1       ;AREA,CHAN,BUF,WDCNT,SECT
;
L4:     BCC     L7              ;NO ERROR IF CARRY CLEAR
;
        MOVB    ERRBYT,R4       ;HOLD ERROR CODE + 1 (TO MAKE NON ZERO)
        INC     R4               ;ERRORS Ø,1,2 GO TO 1,2,3
        MOV     R4,IOERR        ;IN IOERR IN COMMON/FFTCOM/
L7:     RTS     PC
;
L5:     .WRITW  #AREA,R4,R3,R2,R1       ;AREA,CHAN,BUF,WDCNT,SECT
        BR      L4
AREA:   .BLKW   1Ø      ;AREA FOR RT11 I/O MACROS USE
        .ENDC
;
; BELOW, MORE COMPLEX HANDLING OF NWD.LT.256 (PART SECTORS)
L12:    CLC
        ROR     R1              ;SCALE R1/RØ RIGHT ACCORDING TO R5
        ROR     RØ              ;RØ HOLDS FLOW OUT OF R1
        ASL     R5
        BIT     #256.,R5        ;LOOK FOR BIT 256
```

```
        BNE     L2Ø             ;R1/RØ COMPLETE IF FOUND
        BR      L12
;
L2Ø:    SWAB    RØ
        ASL     RØ
;
; NOW HAVE      RØ=BYTE OFFSET IN LOCAL BUFFER
;               R1=SECTOR REQUIRED
;               R2=WORD COUNT
;               R3=USER BUFA ADDRESS
;
; BELOW, TRANSFER BETWEEN LOCAL AND USER BUFFER
        ADD     #JBUF,RØ        ;RØ=JBUF ADDRESS + OFFSET
        CMP     SECHLD,R1               ;CHECK CURRENT SECTOR LOADED
        BNE     L24             ;MUST FIRST LOAD SECTO AT L24 IF NEC
L21:    TST     JSW             ;TEST WHETHER READ/WRITE
        BLT     L23
;
; BELOW, 'READ' PART SECTOR
L22:    MOV     (RØ)+,(R3)+     ;COPY LOCAL TO USER BUFA
        DEC     R2
        BGT     L22
        JMP     L6              ;FINISH
;
; BELOW, 'WRITE' PART SECTOR
L23:    MOV     (R3)+,(RØ)+     ;COPY USER TO LOCAL BUF
        DEC     R2
        BGT     L23
        MOV     #-1,JSWIO               ;SET UP I/O FOR WRITE
        JSR     PC,LINOUT       ;WRITE FROM LOCAL JBUF (SLOW BUT SAFE)
        JMP     L6              ;FINISH
;
; BELOW, LOAD LOCAL JBUF, IF POSSIBLE
L24:    MOV     R1,SECHLD       ;HOLD CURRENT SECTOR NUM
        MOV     #1,JSWIO        ;SET UP FOR READ
        JSR     PC,LINOUT       ;DO LOCAL READ TO JBUF
        BR      L21
;
; BELOW, LOCAL INPUT/OUTPUT ROUTINE
LINOUT: MOV     RØ,-(SP)        ;SAVE REGISTERS RØ TO R3
        MOV     R1,-(SP)
        MOV     R2,-(SP)
        MOV     R3,-(SP)
        MOV     #256.,R2        ;R2=256. FOR WHOLE SECTOR
        MOV     #JBUF,R3        ;R3 IS LOCAL JBUF ADDRESS
        JSR     PC,INOUT        ;CALL INOUT ROUTINE
        MOV     (SP)+,R3        ;RESTORE REGISTERS RØ TO R3
        MOV     (SP)+,R2
        MOV     (SP)+,R1
        MOV     (SP)+,RØ
        RTS     PC
;
; BELOW, LOCAL VARIABLES
JSW:    .WORD   Ø       ;JSW=+1 MFREAD, -1 MFWRIT
JSWIO:  .WORD   Ø       ;ACTUAL JSW USED FOR INOUT ROUTINE
SECHLD: .WORD   -1      ;CURRENT SECTOR LOADED IN JBUF
JBUF:   .BLKW   256.    ;JBUF 256 WORD LOCAL BUFFER FOR NWD.LT.256
;
        .END
```

ALGORITHM 546
SOLVEBLOK [F4]

CARL DE BOOR
Mathematics Research Center, The University of Wisconsin–Madison
and
RICHARD WEISS
Technische Universität Wien, Austria

Key Words and Phrases: almost block diagonal systems, Gaussian elimination, spline approximation, ordinary differential equations
CR Categories: 5.13, 5.14, 5.17
Language: Fortran

DESCRIPTION

This Fortran package is a complement to [1] where its design is explained and it is compared with related algorithms.

REFERENCES

1. DE BOOR, C., AND WEISS, R. SOLVEBLOK: A package for solving almost block diagonal linear systems. *ACM Trans. Math. Software 6,* 1 (March 1980), 80–87.

ALGORITHM

```
      SUBROUTINE SLVBLK ( BLOKS, INTEGS, NBLOKS, B, IPIVOT, X, IFLAG )  BLKØØ1ØØ
C     THIS PROGRAM SOLVES  THE  LINEAR SYSTEM  A*X = B  WHERE A IS AN    BLKØØ2ØØ
C  ALMOST BLOCK DIAGONAL MATRIX.  SUCH ALMOST BLOCK DIAGONAL MATRICES    BLKØØ3ØØ
C  ARISE NATURALLY IN PIECEWISE POLYNOMIAL INTERPOLATION OR APPROX-      BLKØØ4ØØ
C  IMATION AND IN FINITE ELEMENT METHODS FOR TWO-POINT BOUNDARY VALUE    BLKØØ5ØØ
C  PROBLEMS.  THE PLU FACTORIZATION METHOD IS IMPLEMENTED HERE TO TAKE   BLKØØ6ØØ
C  ADVANTAGE OF THE SPECIAL STRUCTURE OF SUCH SYSTEMS FOR SAVINGS IN     BLKØØ7ØØ
C  COMPUTING TIME AND STORAGE REQUIREMENTS.                              BLKØØ8ØØ
C                                                                        BLKØØ9ØØ
C                  PARAMETERS                                            BLKØ1ØØØ
C  BLOKS   A ONE-DIMENSIONAL ARRAY, OF LENGTH                           BLKØ11ØØ
C                   SUM( INTEGS(1,I)*INTEGS(2,I) , I = 1,NBLOKS )       BLKØ12ØØ
C          ON INPUT, CONTAINS THE BLOCKS OF THE ALMOST BLOCK DIAGONAL    BLKØ13ØØ
C          MATRIX  A  .  THE ARRAY INTEGS (SEE BELOW AND THE EXAMPLE)    BLKØ14ØØ
C          DESCRIBES THE BLOCK STRUCTURE.                                BLKØ15ØØ
C          ON OUTPUT, CONTAINS CORRESPONDINGLY THE PLU FACTORIZATION     BLKØ16ØØ
C          OF  A  (IF IFLAG .NE. Ø).  CERTAIN OF THE ENTRIES INTO BLOKS  BLKØ17ØØ
C          ARE ARBITRARY (WHERE THE BLOCKS OVERLAP).                     BLKØ18ØØ
C  INTEGS  INTEGER ARRAY DESCRIPTION OF THE BLOCK STRUCTURE OF  A .      BLKØ19ØØ
C            INTEGS(1,I) = NO. OF ROWS OF BLOCK I        =  NROW        BLKØ2ØØØ
C            INTEGS(2,I)= NO. OF COLUMNS OF BLOCK I      =  NCOL        BLKØ21ØØ
C            INTEGS(3,I) = NO. OF ELIM. STEPS IN BLOCK I =  LAST        BLKØ22ØØ
```

Received 6 June 1977 and 1 January 1979.

This work was supported by the U.S. Army under Contract DAAG29-75-C-0024.
Authors' addresses: C. de Boor, Mathematics Research Center, University of Wisconsin-Madison, Madison, WI 53706; R. Weiss, Institut für Numerische Mathematik, Technische Universität-Wien, Gusshausstrasse 27-29, A-1040 Wien, Austria.

```
C                        I  = 1,2,...,NBLOKS                                BLKØ23ØØ
C          THE LINEAR SYSTEM IS OF ORDER                                    BLKØ24ØØ
C               N  =  SUM ( INTEGS(3,I) , I=1,...,NBLOKS ),                 BLKØ25ØØ
C          BUT THE TOTAL NUMBER OF ROWS IN THE BLOCKS IS                    BLKØ26ØØ
C              NBROWS = SUM( INTEGS(1,I) , I = 1,...,NBLOKS)                BLKØ27ØØ
C  NBLOKS  NUMBER OF BLOCKS                                                 BLKØ28ØØ
C  B       RIGHT SIDE OF THE LINEAR SYSTEM, ARRAY OF LENGTH NBROWS.         BLKØ29ØØ
C          CERTAIN OF THE ENTRIES ARE ARBITRARY, CORRESPONDING TO           BLKØ3ØØØ
C          ROWS OF THE BLOCKS WHICH OVERLAP (SEE BLOCK STRUCTURE AND        BLKØ31ØØ
C          THE EXAMPLE BELOW).                                              BLKØ32ØØ
C  IPIVOT  ON OUTPUT, INTEGER ARRAY CONTAINING THE PIVOTING SEQUENCE        BLKØ33ØØ
C          USED. LENGTH IS NBROWS                                           BLKØ34ØØ
C  X       ON OUTPUT, CONTAINS THE COMPUTED SOLUTION (IF IFLAG .NE. Ø)      BLKØ35ØØ
C          LENGTH IS N.                                                     BLKØ36ØØ
C  IFLAG   ON OUTPUT, INTEGER                                               BLKØ37ØØ
C            = (-1)**(NO. OF INTERCHANGES DURING FACTORIZATION)             BLKØ38ØØ
C                  IF  A  IS INVERTIBLE                                     BLKØ39ØØ
C            = Ø     IF  A  IS SINGULAR                                     BLKØ4ØØØ
C                                                                           BLKØ41ØØ
C                    AUXILIARY PROGRAMS                                     BLKØ42ØØ
C  FCBLOK (BLOKS,INTEGS,NBLOKS,IPIVOT,SCRTCH,IFLAG)  FACTORS THE MATRIX     BLKØ43ØØ
C           A , AND IS USED FOR THIS PURPOSE IN SLVBLK. ITS ARGUMENTS       BLKØ44ØØ
C          ARE AS IN SLVBLK, EXCEPT FOR                                     BLKØ45ØØ
C              SCRTCH = A WORK ARRAY OF LENGTH MAX(INTEGS(1,I)).            BLKØ46ØØ
C                                                                           BLKØ47ØØ
C  SBBLOK (BLOKS,INTEGS,NBLOKS,IPIVOT,B,X)  SOLVES THE SYSTEM A*X = B       BLKØ48ØØ
C          ONCE  A  IS FACTORED. THIS IS DONE AUTOMATICALLY BY SLVBLK       BLKØ49ØØ
C          FOR ONE RIGHT SIDE B, BUT SUBSEQUENT SOLUTIONS MAY BE            BLKØ5ØØØ
C          OBTAINED FOR ADDITIONAL B-VECTORS. THE ARGUMENTS ARE ALL         BLKØ51ØØ
C          AS IN SLVBLK.                                                    BLKØ52ØØ
C                                                                           BLKØ53ØØ
C  DTBLOK (BLOKS,INTEGS,NBLOKS,IPIVOT,IFLAG,DETSGN,DETLOG) COMPUTES THE     BLKØ54ØØ
C          DETERMINANT OF  A  ONCE SLVBLK OR FCBLOK HAS DONE THE FACT-      BLKØ55ØØ
C          ORIZATION.THE FIRST FIVE ARGUMENTS ARE AS IN SLVBLK.             BLKØ56ØØ
C              DETSGN  = SIGN OF THE DETERMINANT                            BLKØ57ØØ
C              DETLOG  = NATURAL LOG OF THE DETERMINANT                     BLKØ58ØØ
C                                                                           BLKØ59ØØ
C                ------ BLOCK STRUCTURE OF  A  ------                       BLKØ6ØØØ
C  THE NBLOKS BLOCKS ARE STORED CONSECUTIVELY IN THE ARRAY  BLOKS .         BLKØ61ØØ
C  THE FIRST BLOCK HAS ITS (1,1)-ENTRY AT BLOKS(1), AND, IF THE I-TH        BLKØ62ØØ
C  BLOCK HAS ITS (1,1)-ENTRY AT BLOKS(INDEX(I)), THEN                       BLKØ63ØØ
C         INDEX(I+1) = INDEX(I)  +  NROW(I)*NCOL(I) .                       BLKØ64ØØ
C    THE BLOCKS ARE PIECED TOGETHER TO GIVE THE INTERESTING PART OF  A      BLKØ65ØØ
C  AS FOLLOWS.  FOR I = 1,2,...,NBLOKS-1, THE (1,1)-ENTRY OF THE NEXT       BLKØ66ØØ
C  BLOCK (THE (I+1)ST BLOCK ) CORRESPONDS TO THE (LAST+1,LAST+1)-ENTRY      BLKØ67ØØ
C  OF THE CURRENT I-TH BLOCK.  RECALL LAST = INTEGS(3,I) AND NOTE THAT      BLKØ68ØØ
C  THIS MEANS THAT                                                          BLKØ69ØØ
C      A. EVERY BLOCK STARTS ON THE DIAGONAL OF  A .                        BLKØ7ØØØ
C      B. THE BLOCKS OVERLAP (USUALLY). THE ROWS OF THE (I+1)ST BLOCK       BLKØ71ØØ
C         WHICH ARE OVERLAPPED BY THE I-TH BLOCK MAY BE ARBITRARILY DE-     BLKØ72ØØ
C         FINED INITIALLY. THEY ARE OVERWRITTEN DURING ELIMINATION.         BLKØ73ØØ
C    THE RIGHT SIDE FOR THE EQUATIONS IN THE I-TH BLOCK ARE STORED COR-     BLKØ74ØØ
C  RESPONDINGLY AS THE LAST ENTRIES OF A PIECE OF  B  OF LENGTH  NROW       BLKØ75ØØ
C  (= INTEGS(1,I)) AND FOLLOWING IMMEDIATELY IN  B  THE CORRESPONDING       BLKØ76ØØ
C  PIECE FOR THE RIGHT SIDE OF THE PRECEDING BLOCK, WITH THE RIGHT SIDE     BLKØ77ØØ
C  FOR THE FIRST BLOCK STARTING AT  B(1) . IN THIS, THE RIGHT SIDE FOR      BLKØ78ØØ
C  AN EQUATION NEED ONLY BE SPECIFIED ONCE ON INPUT, IN THE FIRST BLOCK     BLKØ79ØØ
C  IN WHICH THE EQUATION APPEARS.                                           BLKØ8ØØØ
C                                                                           BLKØ81ØØ
C                ------ EXAMPLE AND TEST DRIVER ------                     BLKØ82ØØ
C    THE TEST DRIVER FOR THIS PACKAGE CONTAINS AN EXAMPLE, A LINEAR         BLKØ83ØØ
C  SYSTEM OF ORDER 11, WHOSE NONZERO ENTRIES ARE INDICATED IN THE FOL-      BLKØ84ØØ
C  LOWING SCHEMA BY THEIR ROW AND COLUMN INDEX MODULO 1Ø. NEXT TO IT        BLKØ85ØØ
C  ARE THE CONTENTS OF THE  INTEGS  ARRRAY WHEN THE MATRIX IS TAKEN TO      BLKØ86ØØ
C  BE ALMOST BLOCK DIAGONAL WITH  NBLOKS = 5, AND BELOW IT ARE THE FIVE     BLKØ87ØØ
C  BLOCKS.                                                                  BLKØ88ØØ
C                                                                           BLKØ89ØØ
C                       NROW1 = 3, NCOL1 = 4                                BLKØ9ØØØ
C            11 12 13 14                                                    BLKØ91ØØ
C            21 22 23 24    NROW2 = 3, NCOL2 = 3                            BLKØ92ØØ
C            31 32 33 34                                                    BLKØ93ØØ
C  LAST1 = 2      43 44 45                                                 BLKØ94ØØ
C                  53 54 55             NROW3 = 3, NCOL3 = 4               BLKØ95ØØ
C        LAST2 = 3          66 67 68 69    NROW4 = 3, NCOL4 = 4            BLKØ96ØØ
C                           76 77 78 79       NROW5 = 4, NCOL5 = 4         BLKØ97ØØ
C                           86 87 88 89                                     BLKØ98ØØ
C                 LAST3 = 1   97 98 99 9Ø                                   BLKØ99ØØ
```

```
C                    LAST4 = 1   Ø8 Ø9 ØØ Ø1                                BLK1ØØØØ
C                                18 19 1Ø 11                                BLK1Ø1ØØ
C                       LAST5 = 4                                           BLK1Ø2ØØ
C                                                                           BLK1Ø3ØØ
C         ACTUAL INPUT TO BLOKS SHOWN BY ROWS OF BLOCKS OF  A .             BLK1Ø4ØØ
C       (THE ** ITEMS ARE ARBITRARY, THIS STORAGE IS USED BY SLVBLK)        BLK1Ø5ØØ
C                                                                           BLK1Ø6ØØ
C  11 12 13 14  / ** ** **  / 66 67 68 69  / ** ** ** **  / ** ** ** **  BLK1Ø7ØØ
C  21 22 23 24 /  43 44 45 /  76 77 78 79 /  ** ** ** ** /  ** ** ** **  BLK1Ø8ØØ
C  31 32 33 34/   53 54 55/   86 87 88 89/   97 98 99 9Ø/   Ø8 Ø9 ØØ Ø1  BLK1Ø9ØØ
C                                                          18 19 1Ø 11  BLK11ØØØ
C                                                                           BLK111ØØ
C  INDEX = 1      INDEX = 13  INDEX = 22     INDEX = 34     INDEX = 46     BLK112ØØ
C                                                                           BLK113ØØ
C         ACTUAL RIGHT SIDE VALUES WITH ** FOR ARBITRARY VALUES             BLK114ØØ
C  B1 B2 B3 ** B4 B5 B6 B7 B8 ** ** B9 ** ** B1Ø B11                        BLK115ØØ
C                                                                           BLK116ØØ
C  (IT WOULD HAVE BEEN MORE EFFICIENT TO COMBINE BLOCK 3 WITH BLOCK 4)      BLK117ØØ
C                                                                           BLK118ØØ
      INTEGER INTEGS(3,NBLOKS),IPIVOT(1),IFLAG                              BLK119ØØ
      REAL BLOKS(1),B(1),X(1)                                               BLK12ØØØ
C     IN THE CALL TO FCBLOK,  X  IS USED FOR TEMPORARY STORAGE.             BLK121ØØ
      CALL FCBLOK(BLOKS,INTEGS,NBLOKS,IPIVOT,X,IFLAG)                       BLK122ØØ
      IF (IFLAG .EQ. Ø)                    RETURN                           BLK123ØØ
      CALL SBBLOK(BLOKS,INTEGS,NBLOKS,IPIVOT,B,X)                           BLK124ØØ
                                           RETURN                           BLK125ØØ
      END                                                                   BLK126ØØ


      SUBROUTINE FCBLOK ( BLOKS, INTEGS, NBLOKS, IPIVOT, SCRTCH, IFLAG )BLK127ØØ
CALLS SUBROUTINES  F A C T R B  AND  S H I F T B .                          BLK128ØØ
C                                                                           BLK129ØØ
C   F C B L O K  SUPERVISES THE PLU FACTORIZATION WITH PIVOTING OF          BLK13ØØØ
C  SCALED ROWS OF THE ALMOST BLOCK DIAGONAL MATRIX STORED IN THE ARRAYS     BLK131ØØ
C   B L O K S  AND  I N T E G S .                                           BLK132ØØ
C                                                                           BLK133ØØ
C   FACTRB = SUBPROGRAM WHICH CARRIES OUT STEPS 1,...,LAST OF GAUSS         BLK134ØØ
C            ELIMINATION (WITH PIVOTING) FOR AN INDIVIDUAL BLOCK.           BLK135ØØ
C   SHIFTB = SUBPROGRAM WHICH SHIFTS THE REMAINING ROWS TO THE TOP OF       BLK136ØØ
C            THE NEXT BLOCK                                                 BLK137ØØ
C                                                                           BLK138ØØ
C PARAMETERS                                                                BLK139ØØ
C    BLOKS   AN ARRAY THAT INITIALLY CONTAINS THE ALMOST BLOCK DIAGONAL     BLK14ØØØ
C            MATRIX  A  TO BE FACTORED, AND ON RETURN CONTAINS THE COM-     BLK141ØØ
C            PUTED FACTORIZATION OF  A .                                    BLK142ØØ
C    INTEGS  AN INTEGER ARRAY DESCRIBING THE BLOCK STRUCTURE OF  A .        BLK143ØØ
C    NBLOKS  THE NUMBER OF BLOCKS IN  A .                                   BLK144ØØ
C    IPIVOT  AN INTEGER ARRAY OF DIMENSION  SUM (INTEGS(1,I) , I=1,         BLK145ØØ
C            ...,NBLOKS) WHICH, ON RETURN, CONTAINS THE PIVOTING STRA-      BLK146ØØ
C            TEGY USED.                                                     BLK147ØØ
C    SCRTCH  WORK AREA REQUIRED, OF LENGTH  MAX (INTEGS(1,I) , I=1,         BLK148ØØ
C            ...,NBLOKS).                                                   BLK149ØØ
C    IFLAG   OUTPUT PARAMETER,                                              BLK15ØØØ
C            = Ø  IN CASE MATRIX WAS FOUND TO BE SINGULAR.                  BLK151ØØ
C            OTHERWISE,                                                     BLK152ØØ
C            = (-1)**(NUMBER OF ROW INTERCHANGES DURING FACTORIZATION)      BLK153ØØ
C                                                                           BLK154ØØ
      INTEGER INTEGS(3,NBLOKS),IPIVOT(1),IFLAG, I,INDEX,INDEXB,INDEXN,      BLK155ØØ
     *        LAST,NCOL,NROW                                                BLK156ØØ
      REAL BLOKS(1),SCRTCH(1)                                               BLK157ØØ
      IFLAG = 1                                                             BLK158ØØ
      INDEXB = 1                                                            BLK159ØØ
      INDEXN = 1                                                            BLK16ØØØ
      I = 1                                                                 BLK161ØØ
C                          LOOP OVER THE BLOCKS.  I  IS LOOP INDEX          BLK162ØØ
   1Ø    INDEX = INDEXN                                                     BLK163ØØ
         NROW = INTEGS(1,I)                                                 BLK164ØØ
         NCOL = INTEGS(2,I)                                                 BLK165ØØ
         LAST = INTEGS(3,I)                                                 BLK166ØØ
C        CARRY OUT ELIMINATION ON THE I-TH BLOCK UNTIL NEXT BLOCK           BLK167ØØ
C        ENTERS, I.E., FOR COLUMNS 1,...,LAST  OF I-TH BLOCK.               BLK168ØØ
         CALL FACTRB(BLOKS(INDEX),IPIVOT(INDEXB),SCRTCH,NROW,NCOL,LAST,     BLK169ØØ
     *              IFLAG)                                                  BLK17ØØØ
C         CHECK FOR HAVING REACHED A SINGULAR BLOCK OR THE LAST BLOCK       BLK171ØØ
         IF (IFLAG .EQ. Ø .OR. I .EQ. NBLOKS)                               BLK172ØØ
     *                                  RETURN                             BLK173ØØ
```

```
      I = I+1                                                          BLK174ØØ
      INDEXN = NROW*NCOL + INDEX                                       BLK175ØØ
C           PUT THE REST OF THE I-TH BLOCK ONTO THE NEXT BLOCK         BLK176ØØ
      CALL SHIFTB(BLOKS(INDEX),IPIVOT(INDEXB),NROW,NCOL,LAST,          BLK177ØØ
     *          BLOKS(INDEXN),INTEGS(1,I),INTEGS(2,I))                 BLK178ØØ
      INDEXB = INDEXB + NROW                                           BLK179ØØ
                                    GO TO 1Ø                           BLK18ØØØ
      END                                                              BLK181ØØ


      SUBROUTINE FACTRB ( W, IPIVOT, D, NROW, NCOL, LAST, IFLAG )      BLK182ØØ
C  ADAPTED FROM P.132 OF *ELEMENTARY NUMER.ANALYSIS* BY CONTE-DE BOOR  BLK183ØØ
C                                                                      BLK184ØØ
C  CONSTRUCTS A PARTIAL PLU FACTORIZATION, CORRESPONDING TO STEPS 1,...,BLK185ØØ
C   L A S T   IN GAUSS ELIMINATION, FOR THE MATRIX  W  OF ORDER        BLK186ØØ
C   ( N R O W ,  N C O L ), USING PIVOTING OF SCALED ROWS.             BLK187ØØ
C                                                                      BLK188ØØ
C  PARAMETERS                                                          BLK189ØØ
C    W       CONTAINS THE (NROW,NCOL) MATRIX TO BE PARTIALLY FACTORED  BLK19ØØØ
C            ON INPUT, AND THE PARTIAL FACTORIZATION ON OUTPUT.        BLK191ØØ
C    IPIVOT  AN INTEGER ARRAY OF LENGTH NROW CONTAINING A RECORD OF THE BLK192ØØ
C            PIVOTING STRATEGY USED. ROW IPIVOT(I) IS USED DURING THE  BLK193ØØ
C            I-TH ELIMINATION STEP, I=1,...,LAST.                      BLK194ØØ
C    D       A WORK ARRAY OF LENGTH NROW USED TO STORE ROW SIZES        BLK195ØØ
C            TEMPORARILY.                                              BLK196ØØ
C    NROW    NUMBER OF ROWS OF W.                                      BLK197ØØ
C    NCOL    NUMBER OF COLUMNS OF W.                                   BLK198ØØ
C    LAST    NUMBER OF ELIMINATION STEPS TO BE CARRIED OUT.            BLK199ØØ

C    IFLAG   ON OUTPUT, EQUALS IFLAG ON INPUT TIMES (-1)**(NUMBER OF   BLK2ØØØØ
C            ROW INTERCHANGES DURING THE FACTORIZATION PROCESS), IN    BLK2Ø1ØØ
C            CASE NO ZERO PIVOT WAS ENCOUNTERED.                       BLK2Ø2ØØ
C            OTHERWISE, IFLAG = Ø ON OUTPUT.                           BLK2Ø3ØØ
C                                                                      BLK2Ø4ØØ
      INTEGER IPIVOT(NROW),NCOL,LAST,IFLAG, I,IPIVI,IPIVK,J,K,KP1      BLK2Ø5ØØ
      REAL W(NROW,NCOL),D(NROW), AWIKDI,COLMAX,RATIO,ROWMAX            BLK2Ø6ØØ
C  INITIALIZE IPIVOT, D                                                BLK2Ø7ØØ
      DO 1Ø I=1,NROW                                                   BLK2Ø8ØØ
         IPIVOT(I) = I                                                 BLK2Ø9ØØ
         ROWMAX = Ø.                                                   BLK21ØØØ
         DO 9 J=1,NCOL                                                 BLK211ØØ
    9       ROWMAX = AMAX1(ROWMAX, ABS(W(I,J)))                        BLK212ØØ
         IF (ROWMAX .EQ. Ø.)             GO TO 999                     BLK213ØØ
   1Ø    D(I) = ROWMAX                                                 BLK214ØØ
C GAUSS ELIMINATION WITH PIVOTING OF SCALED ROWS, LOOP OVER K=1,.,LAST BLK215ØØ
      K = 1                                                            BLK216ØØ
C        AS PIVOT ROW FOR K-TH STEP, PICK AMONG THE ROWS NOT YET USED,  BLK217ØØ
C        I.E., FROM ROWS IPIVOT(K),...,IPIVOT(NROW), THE ONE WHOSE K-TH BLK218ØØ
C        ENTRY (COMPARED TO THE ROW SIZE) IS LARGEST. THEN, IF THIS ROW BLK219ØØ
C        DOES NOT TURN OUT TO BE ROW IPIVOT(K), REDEFINE IPIVOT(K) AP-  BLK22ØØØ
C        PROPRIATELY AND RECORD THIS INTERCHANGE BY CHANGING THE SIGN   BLK221ØØ
C        OF  I F L A G .                                               BLK222ØØ
   11    IPIVK = IPIVOT(K)                                             BLK223ØØ
         IF (K .EQ. NROW)                GO TO 21                      BLK224ØØ
         J = K                                                         BLK225ØØ
         KP1 = K+1                                                     BLK226ØØ
         COLMAX = ABS(W(IPIVK,K))/D(IPIVK)                             BLK227ØØ
C             FIND THE (RELATIVELY) LARGEST PIVOT                      BLK228ØØ
         DO 15 I=KP1,NROW                                              BLK229ØØ
            IPIVI = IPIVOT(I)                                          BLK23ØØØ
            AWIKDI = ABS(W(IPIVI,K))/D(IPIVI)                          BLK231ØØ
            IF (AWIKDI .LE. COLMAX)      GO TO 15                      BLK232ØØ
               COLMAX = AWIKDI                                         BLK233ØØ
               J = I                                                   BLK234ØØ
   15       CONTINUE                                                  BLK235ØØ
         IF (J .EQ. K)                   GO TO 16                      BLK236ØØ
         IPIVK = IPIVOT(J)                                             BLK237ØØ
         IPIVOT(J) = IPIVOT(K)                                         BLK238ØØ
         IPIVOT(K) = IPIVK                                             BLK239ØØ
         IFLAG = -IFLAG                                                BLK24ØØØ
   16    CONTINUE                                                      BLK241ØØ
C        IF PIVOT ELEMENT IS TOO SMALL IN ABSOLUTE VALUE, DECLARE      BLK242ØØ
C        MATRIX TO BE NONINVERTIBLE AND QUIT.                          BLK243ØØ
         IF (ABS(W(IPIVK,K))+D(IPIVK) .LE. D(IPIVK))                   BLK244ØØ
     *                                  GO TO 999                     BLK245ØØ
C        OTHERWISE, SUBTRACT THE APPROPRIATE MULTIPLE OF THE PIVOT     BLK246ØØ
```

```
C        ROW FROM REMAINING ROWS, I.E., THE ROWS IPIVOT(K+1),...,       BLK247ØØ
C        IPIVOT(NROW), TO MAKE K-TH ENTRY ZERO. SAVE THE MULTIPLIER IN  BLK248ØØ
C        ITS PLACE.                                                     BLK249ØØ
         DO 2Ø I=KP1,NROW                                               BLK25ØØØ
            IPIVI = IPIVOT(I)                                           BLK251ØØ
            W(IPIVI,K) = W(IPIVI,K)/W(IPIVK,K)                          BLK252ØØ
            RATIO = -W(IPIVI,K)                                         BLK253ØØ
            DO 2Ø J=KP1,NCOL                                            BLK254ØØ
   2Ø          W(IPIVI,J) = RATIO*W(IPIVK,J) + W(IPIVI,J)               BLK255ØØ
         K = KP1                                                        BLK256ØØ
C        CHECK FOR HAVING REACHED THE NEXT BLOCK.                       BLK257ØØ
         IF (K .LE. LAST)                GO TO 11                       BLK258ØØ
                                         RETURN                         BLK259ØØ
C     IF  LAST  .EQ. NROW , CHECK NOW THAT PIVOT ELEMENT IN LAST ROW    BLK26ØØØ
C     IS NONZERO.                                                       BLK261ØØ
   21 IF( ABS(W(IPIVK,NROW))+D(IPIVK) .GT. D(IPIVK) )                   BLK262ØØ
     *                                   RETURN                         BLK263ØØ
C                  SINGULARITY FLAG SET                                BLK264ØØ
  999 IFLAG = Ø                                                         BLK265ØØ
                                         RETURN                         BLK266ØØ
      END                                                               BLK267ØØ


      SUBROUTINE SHIFTB ( AI, IPIVOT, NROWI, NCOLI, LAST,               BLK268ØØ
     *                    AI1, NROWI1, NCOLI1 )                         BLK269ØØ
C  SHIFTS THE ROWS IN CURRENT BLOCK, AI, NOT USED AS PIVOT ROWS, IF     BLK27ØØØ
C  ANY, I.E., ROWS IPIVOT(LAST+1),...,IPIVOT(NROWI), ONTO THE FIRST    BLK271ØØ
C  MMAX = NROW-LAST ROWS OF THE NEXT BLOCK, AI1, WITH COLUMN LAST+J OF BLK272ØØ
C  AI  GOING TO COLUMN J , J=1,...,JMAX=NCOLI-LAST. THE REMAINING COL- BLK273ØØ
C  UMNS OF THESE ROWS OF AI1 ARE ZEROED OUT.                          BLK274ØØ
C                                                                     BLK275ØØ
C                              PICTURE                                BLK276ØØ
C                                                                     BLK277ØØ
C       ORIGINAL SITUATION AFTER          RESULTS IN A NEW BLOCK I+1  BLK278ØØ
C       LAST = 2 COLUMNS HAVE BEEN        CREATED AND READY TO BE     BLK279ØØ
C       DONE IN FACTRB (ASSUMING NO       FACTORED BY NEXT FACTRB CALL. BLK28ØØØ
C       INTERCHANGES OF ROWS)                                         BLK281ØØ
C                  1                                                  BLK282ØØ
C              X  X 1X  X  X          X  X  X  X  X                   BLK283ØØ
C                  1                                                  BLK284ØØ
C              Ø  X 1X  X  X          Ø  X  X  X  X                   BLK285ØØ
C  BLOCK I         1                                                  BLK286ØØ
C  NROWI = 4   Ø  Ø 1X  X  X          Ø  Ø 1X  X  X  Ø  Ø1            BLK287ØØ
C  NCOLI = 5       1                                                  BLK288ØØ
C  LAST = 2    Ø  Ø 1X  X  X          Ø  Ø 1X  X  X  Ø  Ø1            BLK289ØØ
C  -------------------------------          1            1   NEW      BLK29ØØØ
C                  1X  X  X  X  X          1X  X  X  X  X1  BLOCK     BLK291ØØ
C                  1                       1            1   I+1      BLK292ØØ
C  BLOCK I+1       1X  X  X  X  X          1X  X  X  X  X1            BLK293ØØ
C  NROWI1= 5       1                       1            1            BLK294ØØ
C  NCOLI1= 5       1X  X  X  X  X          1X  X  X  X  X1            BLK295ØØ
C  -------------------------------          1-------------1            BLK296ØØ
C                  1                                                  BLK297ØØ
C                                                                     BLK298ØØ
      INTEGER IPIVOT(NROWI),LAST, IP,J,JMAX,JMAXP1,M,MMAX             BLK299ØØ
      REAL AI(NROWI,NCOLI),AI1(NROWI1,NCOLI1)                         BLK3ØØØØ
      MMAX = NROWI - LAST                                             BLK3Ø1ØØ
      JMAX = NCOLI - LAST                                             BLK3Ø2ØØ
      IF (MMAX .LT. 1 .OR. JMAX .LT. 1) RETURN                        BLK3Ø3ØØ
C             PUT THE REMAINDER OF BLOCK I INTO AI1                   BLK3Ø4ØØ
      DO 1Ø M=1,MMAX                                                  BLK3Ø5ØØ
         IP = IPIVOT(LAST+M)                                          BLK3Ø6ØØ
         DO 1Ø J=1,JMAX                                               BLK3Ø7ØØ
   1Ø       AI1(M,J) = AI(IP,LAST+J)                                  BLK3Ø8ØØ
      IF (JMAX .EQ. NCOLI1)           RETURN                          BLK3Ø9ØØ
C             ZERO OUT THE UPPER RIGHT CORNER OF AI1                  BLK31ØØØ
      JMAXP1 = JMAX + 1                                               BLK311ØØ
      DO 2Ø J=JMAXP1,NCOLI1                                           BLK312ØØ
         DO 2Ø M=1,MMAX                                               BLK313ØØ
   2Ø       AI1(M,J) = Ø.                                             BLK314ØØ
                                         RETURN                       BLK315ØØ
      END                                                             BLK316ØØ
```

```
      SUBROUTINE SBBLOK ( BLOKS, INTEGS, NBLOKS, IPIVOT, B, X )          BLK317ØØ
CALLS SUBROUTINES  S U B F O R  AND  S U B B A K .                       BLK318ØØ
C                                                                        BLK319ØØ
C  SUPERVISES THE SOLUTION (BY FORWARD AND BACKWARD SUBSTITUTION) OF     BLK32ØØØ
C  THE LINEAR SYSTEM  A*X = B  FOR X, WITH THE PLU FACTORIZATION OF  A   BLK321ØØ
C  ALREADY GENERATED IN  F C B L O K .  INDIVIDUAL BLOCKS OF EQUATIONS   BLK322ØØ
C  ARE SOLVED VIA  S U B F O R  AND  S U B B A K .                       BLK323ØØ
C                                                                        BLK324ØØ
C PARAMETERS                                                             BLK325ØØ
C    BLOKS, INTEGS, NBLOKS, IPIVOT    ARE AS ON RETURN FROM FCBLOK.      BLK326ØØ
C    B       THE RIGHT SIDE, STORED CORRESPONDING TO THE STORAGE OF      BLK327ØØ
C            THE EQUATIONS. SEE COMMENTS IN  S L V B L K  FOR DETAILS.   BLK328ØØ
C    X       SOLUTION VECTOR                                            BLK329ØØ
C                                                                        BLK33ØØØ
      INTEGER INTEGS(3,NBLOKS),IPIVOT(1), I,INDEX,INDEXB,INDEXX,J,LAST, BLK331ØØ
     *        NBP1,NCOL,NROW                                             BLK332ØØ
      REAL BLOKS(1),B(1),X(1)                                            BLK333ØØ
C                                                                        BLK334ØØ
C      FORWARD SUBSTITUTION PASS                                         BLK335ØØ
C                                                                        BLK336ØØ
      INDEX = 1                                                          BLK337ØØ
      INDEXB = 1                                                         BLK338ØØ
      INDEXX = 1                                                         BLK339ØØ
      DO 2Ø I=1,NBLOKS                                                   BLK34ØØØ
         NROW = INTEGS(1,I)                                              BLK341ØØ
         LAST = INTEGS(3,I)                                              BLK342ØØ
         CALL SUBFOR(BLOKS(INDEX),IPIVOT(INDEXB),NROW,LAST,B(INDEXB),    BLK343ØØ
     *               X(INDEXX))                                          BLK344ØØ
         INDEX = NROW*INTEGS(2,I) + INDEX                                BLK345ØØ
         INDEXB = INDEXB + NROW                                          BLK346ØØ
   2Ø    INDEXX = INDEXX + LAST                                          BLK347ØØ
C                                                                        BLK348ØØ
C     BACK SUBSTITUTION PASS                                             BLK349ØØ
C                                                                        BLK35ØØØ
      NBP1 = NBLOKS + 1                                                  BLK351ØØ
      DO 3Ø J=1,NBLOKS                                                   BLK352ØØ
         I = NBP1 - J                                                    BLK353ØØ
         NROW = INTEGS(1,I)                                              BLK354ØØ
         NCOL = INTEGS(2,I)                                              BLK355ØØ
         LAST = INTEGS(3,I)                                              BLK356ØØ
         INDEX = INDEX - NROW*NCOL                                       BLK357ØØ
         INDEXB = INDEXB - NROW                                          BLK358ØØ
         INDEXX = INDEXX - LAST                                          BLK359ØØ
   3Ø    CALL SUBBAK(BLOKS(INDEX),IPIVOT(INDEXB),NROW,NCOL,LAST,         BLK36ØØØ
     *               X(INDEXX))                                          BLK361ØØ
                                      RETURN                             BLK362ØØ
      END                                                                BLK363ØØ


      SUBROUTINE SUBFOR ( W, IPIVOT, NROW, LAST, B, X )                  BLK364ØØ
C  CARRIES OUT THE FORWARD PASS OF SUBSTITUTION FOR THE CURRENT BLOCK,    BLK365ØØ
C  I.E., THE ACTION ON THE RIGHT SIDE CORRESPONDING TO THE ELIMINATION    BLK366ØØ
C  CARRIED OUT IN  F A C T R B  FOR THIS BLOCK.                          BLK367ØØ
C     AT THE END, X(J) CONTAINS THE RIGHT SIDE OF THE TRANSFORMED        BLK368ØØ
C  IPIVOT(J)-TH EQUATION IN THIS BLOCK, J=1,...,NROW. THEN, SINCE         BLK369ØØ
C  FOR I=1,...,NROW-LAST, B(NROW+I) IS GOING TO BE USED AS THE RIGHT      BLK37ØØØ
C  SIDE OF EQUATION  I  IN THE NEXT BLOCK (SHIFTED OVER THERE FROM        BLK371ØØ
C  THIS BLOCK DURING FACTORIZATION), IT IS SET EQUAL TO X(LAST+I) HERE.   BLK372ØØ
C                                                                        BLK373ØØ
C PARAMETERS                                                             BLK374ØØ
C    W, IPIVOT, NROW, LAST  ARE AS ON RETURN FROM FACTRB.                BLK375ØØ
C    B(J)    IS EXPECTED TO CONTAIN, ON INPUT, THE RIGHT SIDE OF J-TH    BLK376ØØ
C            EQUATION FOR THIS BLOCK, J=1,...,NROW.                      BLK377ØØ
C    B(NROW+J)   CONTAINS, ON OUTPUT, THE APPROPRIATELY MODIFIED RIGHT   BLK378ØØ
C            SIDE FOR EQUATION J IN NEXT BLOCK, J=1,...,NROW-LAST.       BLK379ØØ
C    X(J)    CONTAINS, ON OUTPUT, THE APPROPRIATELY MODIFIED RIGHT       BLK38ØØØ
C            SIDE OF EQUATION IPIVOT(J) IN THIS BLOCK, J=1,...,LAST (AND BLK381ØØ
C            EVEN FOR J=LAST+1,...,NROW).                                BLK382ØØ
C                                                                        BLK383ØØ
      INTEGER IPIVOT(NROW), IP,JMAX,K                                    BLK384ØØ
C     DIMENSION B(NROW + NROW-LAST)                                      BLK385ØØ
      REAL W(NROW,LAST),B(1),X(NROW)                                     BLK386ØØ
      IP = IPIVOT(1)                                                     BLK387ØØ
      X(1) = B(IP)                                                       BLK388ØØ
      IF (NROW .EQ. 1)                  GO TO 99                         BLK389ØØ
      DO 15 K=2,NROW                                                     BLK39ØØØ
```

```
      IP = IPIVOT(K)                                                    BLK391ØØ
      JMAX = AMINØ(K-1,LAST)                                            BLK392ØØ
      SUM = Ø.                                                          BLK393ØØ
      DO 14 J=1,JMAX                                                    BLK394ØØ
14       SUM = W(IP,J)*X(J) + SUM                                       BLK395ØØ
15    X(K) = B(IP) - SUM                                                BLK396ØØ
C                                                                       BLK397ØØ
C     TRANSFER MODIFIED RIGHT SIDES OF EQUATIONS IPIVOT(LAST+1),...,    BLK398ØØ
C     IPIVOT(NROW) TO NEXT BLOCK.                                       BLK399ØØ
      NROWML = NROW - LAST                                              BLK4ØØØØ
      IF (NROWML .EQ. Ø)                GO TO 99                        BLK4Ø1ØØ
      LASTP1 = LAST+1                                                   BLK4Ø2ØØ
      DO 25 K=LASTP1,NROW                                               BLK4Ø3ØØ
25    B(NROWML+K) = X(K)                                                BLK4Ø4ØØ
99                                      RETURN                          BLK4Ø5ØØ
      END                                                               BLK4Ø6ØØ


      SUBROUTINE SUBBAK ( W, IPIVOT, NROW, NCOL, LAST, X )              BLK4Ø7ØØ
C  CARRIES OUT BACKSUBSTITUTION FOR CURRENT BLOCK.                      BLK4Ø8ØØ
C                                                                       BLK4Ø9ØØ
C PARAMETERS                                                            BLK41ØØØ
C    W, IPIVOT, NROW, NCOL, LAST  ARE AS ON RETURN FROM FACTRB.         BLK411ØØ
C    X(1),...,X(NCOL)  CONTAINS, ON INPUT, THE RIGHT SIDE FOR THE       BLK412ØØ
C            EQUATIONS IN THIS BLOCK AFTER BACKSUBSTITUTION HAS BEEN    BLK413ØØ
C            CARRIED UP TO BUT NOT INCLUDING EQUATION IPIVOT(LAST).     BLK414ØØ
C            MEANS THAT X(J) CONTAINS THE RIGHT SIDE OF EQUATION IPI-   BLK415ØØ
C            VOT(J) AS MODIFIED DURING ELIMINATION, J=1,...,LAST, WHILE BLK416ØØ
C            FOR J .GT. LAST, X(J) IS ALREADY A COMPONENT OF THE SOLUT- BLK417ØØ
C            ION VECTOR.                                                BLK418ØØ
C    X(1),...,X(NCOL) CONTAINS, ON OUTPUT, THE COMPONENTS OF THE SOLUT- BLK419ØØ
C            ION CORRESPONDING TO THE PRESENT BLOCK.                    BLK42ØØØ
C                                                                       BLK421ØØ
      INTEGER IPIVOT(NROW),LAST,  IP,J,K,KP1                            BLK422ØØ
      REAL W(NROW,NCOL),X(NCOL), SUM                                    BLK423ØØ
      K = LAST                                                          BLK424ØØ
      IP = IPIVOT(K)                                                    BLK425ØØ
      SUM = Ø.                                                          BLK426ØØ
      IF (K .EQ. NCOL)                  GO TO 4                         BLK427ØØ
      KP1 = K+1                                                         BLK428ØØ
2     DO 3 J=KP1,NCOL                                                   BLK429ØØ
3        SUM = W(IP,J)*X(J) + SUM                                       BLK43ØØØ
4     X(K) = (X(K) - SUM)/W(IP,K)                                       BLK431ØØ
         IF (K .EQ. 1)                  RETURN                         BLK432ØØ
         KP1 = K                                                        BLK433ØØ
         K = K-1                                                        BLK434ØØ
         IP = IPIVOT(K)                                                 BLK435ØØ
         SUM = Ø.                                                       BLK436ØØ
                                        GO TO 2                         BLK437ØØ
      END                                                               BLK438ØØ


      SUBROUTINE DTBLOK ( BLOKS, INTEGS, NBLOKS, IPIVOT, IFLAG,         BLK439ØØ
     *                    DETSGN, DETLOG )                              BLK44ØØØ
C  COMPUTES THE DETERMINANT OF AN ALMOST BLOCK DIAGONAL MATRIX WHOSE    BLK441ØØ
C  PLU FACTORIZATION HAS BEEN OBTAINED PREVIOUSLY IN FCBLOK.            BLK442ØØ
C  *** THE LOGARITHM OF THE DETERMINANT IS COMPUTED INSTEAD OF THE      BLK443ØØ
C  DETERMINANT ITSELF TO AVOID THE DANGER OF OVERFLOW OR UNDERFLOW      BLK444ØØ
C  INHERENT IN THIS CALCULATION.                                        BLK445ØØ
C                                                                       BLK446ØØ
C PARAMETERS                                                            BLK447ØØ
C    BLOKS, INTEGS, NBLOKS, IPIVOT, IFLAG  ARE AS ON RETURN FROM FCBLOK.BLK448ØØ
C            IN PARTICULAR, IFLAG = (-1)**(NUMBER OF INTERCHANGES DUR-  BLK449ØØ
C            ING FACTORIZATION) IF SUCCESSFUL, OTHERWISE IFLAG = Ø.     BLK45ØØØ
C    DETSGN  ON OUTPUT, CONTAINS THE SIGN OF THE DETERMINANT.           BLK451ØØ
C    DETLOG  ON OUTPUT, CONTAINS THE NATURAL LOGARITHM OF THE DETERMI-  BLK452ØØ
C            NANT IF DETERMINANT IS NOT ZERO. OTHERWISE CONTAINS Ø.     BLK453ØØ
C                                                                       BLK454ØØ
      INTEGER INTEGS(3,NBLOKS),IPIVOT(1),IFLAG, I,INDEXP,IP,K,LAST      BLK455ØØ
      REAL BLOKS(1),DETSGN,DETLOG                                       BLK456ØØ
C                                                                       BLK457ØØ
      DETSGN = IFLAG                                                    BLK458ØØ
      DETLOG = Ø.                                                       BLK459ØØ
      IF (IFLAG .EQ. Ø)                 RETURN                          BLK46ØØØ
      INDEX = Ø                                                         BLK461ØØ
```

```
      INDEXP = Ø                                                        BLK462ØØ
      DO 2 I=1,NBLOKS                                                   BLK463ØØ
         NROW = INTEGS(1,I)                                             BLK464ØØ
         LAST = INTEGS(3,I)                                             BLK465ØØ
         DO 1 K=1,LAST                                                  BLK466ØØ
            IP = INDEX + NROW*(K-1) + IPIVOT(INDEXP+K)                  BLK467ØØ
            DETLOG = DETLOG + ALOG(ABS(BLOKS(IP)))                      BLK468ØØ
    1       DETSGN = DETSGN*SIGN(1.,BLOKS(IP))                          BLK469ØØ
         INDEX = NROW*INTEGS(2,I) + INDEX                               BLK47ØØØ
    2    INDEXP = INDEXP + NROW                                         BLK471ØØ
                                      RETURN                            BLK472ØØ
      END                                                               BLK473ØØ


C  TEST PROGRAM FOR THE SOLVEBLOK PACKAGE SOLVES AN 11TH ORDER LINEAR   BLK474ØØ
C  ALMOST BLOCK DIAGONAL SYSTEM.                                        BLK475ØØ
      DIMENSION BLOKS(61),B(16),X(11), IPIVOT(16),INTEGS(15)            BLK476ØØ
C               NROW NCOL LAST                                          BLK477ØØ
      DATA INTEGS/  3 ,  4 ,  2                                         BLK478ØØ
     2           ,  3 ,  3 ,  3                                         BLK479ØØ
     3           ,  3 ,  4 ,  1                                         BLK48ØØØ
     4           ,  3 ,  4 ,  1                                         BLK481ØØ
     5           ,  4 ,  4 ,  4/                                        BLK482ØØ
      DATA BLOKS /.1,.2,-1.,  2.,-.2,.3,  -.1,-.2,-.3,  -.1,4.,.3      BLK483ØØ
     2           ,Ø.,-.4,3.,  Ø.,.4,.5,  Ø.,-5.,-.5                    BLK484ØØ
     3           ,.6,.5,3.,  -.6,4.,.4,  -.6,.5,-.4,  5.,-.5,.4        BLK485ØØ
     4           ,Ø.,Ø.,.3,  Ø.,Ø.,-.3,  Ø.,Ø.,.3,  Ø.,Ø.,7.          BLK486ØØ
     5           ,Ø.,Ø.,.2,6.,  Ø.,Ø.,-.2,.1,  Ø.,Ø.,-.2,-.1,         BLK487ØØ
     5            Ø.,Ø.,8.,-.1/                                        BLK488ØØ
      DATA B /1.94,3.Ø4,-.83,  Ø.,-3.54,2.75,  1.32,2.35,1.96,          BLK489ØØ
     2       2*Ø.,1.52,  2*Ø.,.78,2.4Ø /                               BLK49ØØØ
      NBLOKS = 5                                                        BLK491ØØ
      N = 11                                                            BLK492ØØ
      CALL SLVBLK ( BLOKS, INTEGS, NBLOKS, B, IPIVOT, X, IFLAG )        BLK493ØØ
      ERROR = Ø.                                                        BLK494ØØ
      DO 1Ø I=1,N                                                       BLK495ØØ
         B(I) = FLOAT(12-I)/1Ø. - X(I)                                  BLK496ØØ
   1Ø    ERROR = AMAX1(ERROR,ABS(B(I)))                                 BLK497ØØ
      WRITE (6,61Ø) IFLAG,ERROR, (IPIVOT(I),I=1,16), (I,X(I),B(I),I=1,N)BLK498ØØ
  61Ø FORMAT (28H CHECKOUT SOLVEBLOK ROUTINES/                          BLK499ØØ
     *         8H IFLAG =,I2,17H, MAXIMUM ERROR =,E9.4//               BLK5ØØØØ
     *         9H IPIVOT =,5(2X,3I2),I2//                              BLK5Ø1ØØ
     *        27H  I     X(I)       ERROR(I)/(I3,F11.5,E13.5))        BLK5Ø2ØØ
      CALL DTBLOK ( BLOKS, INTEGS, NBLOKS, IPIVOT, IFLAG,               BLK5Ø3ØØ
     *              DETSGN, DETLOG )                                    BLK5Ø4ØØ
      DET = DETSGN*EXP(DETLOG)                                          BLK5Ø5ØØ
      ERROR = DET - 2.418821899E6                                       BLK5Ø6ØØ
      WRITE (6,611) DET,ERROR                                           BLK5Ø7ØØ
  611 FORMAT(//14H DETERMINANT =,E15.8,11H   ERROR = ,E11.5)            BLK5Ø8ØØ
                                      STOP                              BLK5Ø9ØØ
      END                                                               BLK51ØØØ
```

ALGORITHM 547
Fortran Routines for Discrete Cubic Spline Interpolation and Smoothing [E1], [E3]

CHARLES S. DURIS
Drexel University

Key Words and Phrases: discrete splines, discrete cubic splines, discrete natural splines, interpolation, smoothing
CR Categories: 5.12, 5.13
Language: Fortran

DESCRIPTION

1. Introduction

Two Fortran subroutines, DCSINT and DCSSMO, are presented here for discrete cubic spline interpolation and smoothing. The theory for discrete cubic spline interpolation is given by Lyche in [5, 6]. The theory for discrete natural cubic spline smoothing is given by the author in [2]. For most applications continuous splines (see [3, 4, 7]) are probably more appropriate than discrete splines. Discrete spline interpolation and smoothing are worth considering for problems involving functions defined on discrete equally spaced points, or when difference quotients are available as data rather than derivatives. Possible areas of application are discrete time series analysis, computer routines for plotting data, and actuarial- and demographic-type data analysis.

For both discrete cubic spline interpolation and smoothing we direct our attention to the problem of approximating a function $g(t)$ defined on an interval $[\tau_1, \tau_n]$. The values $g(\tau_i) = g_i$ are specified for $i = 1, 2, \ldots, n$, where $\tau_i < \tau_{i+1}$. For many applications and for deriving the equations in Sections 2 and 3, it is convenient to assume that $g(t)$ is defined on a discrete point set $T_M = \{t_0, t_1, \ldots, t_M\}$ where $t_j = \tau_1 + jh$ for some fixed step size $h > 0$, and also that the τ_i's belong to T_M. Neither the theory nor the routines DCSINT and DCSSMO require that the τ_i's belong to T_M. The only mandatory restriction on the τ_i's is $\tau_i < \tau_{i+1}$.

Discrete cubic splines defined on $[\tau_1, \tau_n]$ with nodes (or knots) $\Lambda_n = \{\tau_1, \tau_2, \ldots, \tau_n\}$ are functions $S(t)$ having the form $S(t) = S_i(t)$ where

$$S_i(t) = g_i + b_i(t - \tau_i) + c_i(t - \tau_i)^2 + d_i(t - \tau_i)^3 \qquad (1)$$

for $t \in [\tau_i, \tau_{i+1}]$ (i.e., $S(t)$ is piecewise cubic). The $S_i(t)$'s satisfy the joining

Received 22 November 1976 and 8 January 1979.

Author's address: Because Dr. Charles Duris has recently passed away, all correspondence regarding this paper should be directed to Dr. Robert Busby, Department of Mathematical Sciences, Drexel University, Philadelphia, PA 19104.

conditions

$$S_i(\tau_{i+1}) = S_{i+1}(\tau_{i+1}), \tag{2}$$

$$\tfrac{1}{2}(\nabla + \Delta)S_i(\tau_{i+1}) = \tfrac{1}{2}(\nabla + \Delta)S_{i+1}(\tau_{i+1}), \tag{3}$$

$$\nabla\Delta S_i(\tau_{i+1}) = \nabla\Delta S_{i+1}(\tau_{i+1}), \tag{4}$$

where $\Delta f(x) = f(x + h) - f(x)$ and $\nabla f(x) = f(x) - f(x - h)$. Conditions (3) and (4) should be recognized as requiring the matching up of the first central and the second central differences at τ_{i+1} for adjacent cubics.

Subroutine DCSINT constructs the discrete cubic spline function on the interval $[\tau_1, \tau_n]$ which interpolates the data (τ_j, g_j) for $j = 1, 2, \ldots, n$ and satisfies one of the three following end conditions:

I. The first central divided difference end conditions

$$\frac{1}{2}\frac{(\nabla + \Delta)S(\tau_1)}{h} = S_1^{(1)} \qquad \text{and} \qquad \frac{1}{2}\frac{(\nabla + \Delta)S(\tau_n)}{h} = S_n^{(1)}$$

for specified values of $S_1^{(1)}$ and $S_n^{(1)}$.

II. The second central divided difference end conditions

$$\frac{\nabla\Delta S(\tau_1)}{h^2} = S_1^{(2)} \qquad \text{and} \qquad \frac{\nabla\Delta S(\tau_n)}{h^2} = S_n^{(2)}$$

for specified values of $S_1^{(2)}$ and $S_n^{(2)}$.

III. The periodic end conditions

$$\tfrac{1}{2}(\nabla + \Delta)S(\tau_1) = \tfrac{1}{2}(\nabla + \Delta)S(\tau_n) \qquad \text{and} \qquad \nabla\Delta S(\tau_1) = \nabla\Delta S(\tau_n)$$

where we assume the data are periodic ($g_1 = g_n$).

In formulating discrete natural cubic spline smoothing, we assume $g(t)$ is defined on a discrete point set T_M. Subroutine DCSSMO constructs the function $S(t)$ which smooths through the data (τ_i, g_i) for $i = 1, 2, \ldots, n$ in the sense that (see [3, 7, 8, 9])

$$\sigma(f(t)) = \rho \sum_{i=1}^{n} W_i[f(\tau_i) - g_i]^2 + \sum_{j=1}^{M-1} [\nabla\Delta f(t_j)]^2 \tag{5}$$

is minimized for $f(t) = S(t)$. This $S(t)$ turns out to be a discrete natural cubic spline ($\nabla\Delta S(\tau_1) = \nabla\Delta S(\tau_n) = 0$). In (5) the W_i's are positive weights specifying the relative importance of the data, and $\rho > 0$ specifies the amount of smoothing in comparison to data fitting. If ρ is small, the smoothness of the approximation is emphasized at the expense of closely fitting the data. If ρ is large, the discrete natural cubic spline more closely fits the data at the expense of smoothness. In particular, as ρ approaches infinity, the smoothing discrete natural spline approaches the interpolating discrete natural spline.

Sections 2 and 3 describe the equations for constructing the discrete cubic splines for interpolating and smoothing, respectively. Those readers mainly interested in seeing how to use subroutines DCSINT and DCSSMO may turn directly to Sections 4 and 5. Section 6 describes how to construct test examples, and Section 7 presents estimates for execution times.

2. Discrete Cubic Spline Interpolation

The discrete cubic spline $S(t) = S_i(t)$ for $t \in [\tau_i, \tau_{i+1}]$ for $i = 1, 2, \ldots, n - 1$ where $S_i(t)$ is given in (1). A better form for deriving equations is to represent $S_i(t)$ using factorial polynomials as follows:

$$\begin{aligned} S_i(t) = {} & g_i + \bar{b}_i(t - \tau_i) + c_i(t - \tau_i)(t - \tau_i - h) \\ & + d_i(t - \tau_i + h)(t - \tau_i)(t - \tau_i - h). \end{aligned} \tag{6}$$

The g_i, c_i, and d_i in (6) are the same as those found in (1), but

$$b_i = \bar{b}_i - c_i h - d_i h^2. \tag{7}$$

The joining conditions (2), (3), and (4) together with $S(\tau_n) = g_n$ give the equations

$$\gamma_i c_i + (\eta_i + \eta_{i+1})c_{i+1} + \gamma_{i+1}c_{i+2} = 3[g_{i+1}^{(1)} - g_i^{(1)}] \tag{8}$$

for $i = 1, 2, \ldots, n-3$, where $H_i = \tau_{i+1} - \tau_i$,

$$\gamma_i = H_i - \frac{h^2}{H_i}, \tag{9}$$

$$\eta_i = 2H_i + \frac{h^2}{H_i}, \tag{10}$$

$$g_i^{(1)} = g[\tau_{i+1}, \tau_i] = \frac{(g_{i+1} - g_i)}{H_i}. \tag{11}$$

The remaining linear equations for the c_i's come from the end conditions. The complete system of linear equations denoted by

$$Ac = g \tag{12}$$

is now described for the three types of end conditions, I, II, and III. These systems are modified forms of the systems found in Lyche [6].

I. The first central divided difference end conditions have

$$\mathrm{A} = \begin{bmatrix} \eta_1 & \gamma_1 & 0 & \cdots & 0 \\ \gamma_1 & (\eta_1 + \eta_2) & \gamma_2 & & \\ 0 & \gamma_2 & (\eta_2 + \eta_3) & \ddots & \\ \vdots & & \ddots & (\eta_{n-2} + \eta_{n-1}) & 0 \\ & & \gamma_{n-2} & & \gamma_{n-1} \\ 0 & \cdots & 0 & \gamma_{n-1} & \eta_{n-1} \end{bmatrix} \tag{13a}$$

$$c = (c_1, c_2, \ldots, c_n)^{\mathrm{T}}, \tag{13b}$$

$$g = 3(g_1^{(1)} - S_1^{(1)}, g_2^{(1)} - g_1^{(1)}, \ldots, g_{n-1}^{(1)} - g_{n-2}^{(1)}, S_n^{(1)} - g_{n-1}^{(1)})^{\mathrm{T}}. \tag{13c}$$

The c_n in (13b) is $\nabla\Delta S(\tau_n)/2h^2$ and is introduced to preserve symmetry.

II. The second central divided difference end conditions have

$$A = \begin{bmatrix} (\eta_1 + \eta_2) & \gamma_2 & 0 & \cdots & 0 \\ \gamma_2 & (\eta_2 + \eta_3) & \gamma_3 & & \vdots \\ 0 & \gamma_3 & & \ddots & \vdots \\ \vdots & & \ddots & & 0 \\ & & & & \gamma_{n-2} \\ 0 & \cdots & 0 & \gamma_{n-2} & (\eta_{n-2} + \eta_{n-1}) \end{bmatrix} \tag{14a}$$

$$\mathrm{c} = (\mathrm{c}_2, \mathrm{c}_3, \ldots, \mathrm{c}_{n-1})^{\mathrm{T}}, \tag{14b}$$

$$g = (g_2^{(1)} - g_1^{(1)} - \tfrac{1}{6}\gamma_1 S_1^{(2)}, g_3^{(1)} - g_2^{(1)}, \ldots, g^{(1)} - g^{(1)} - \tfrac{1}{6}\gamma_{n-1}S^{(2)}). \tag{14c}$$

In [2] it is shown that the natural end conditions $S_1^{(2)} = S_n^{(2)} = 0$ produce the function which interpolates the given data and minimize $\sum_{i=1}^{N-1} [\nabla\Delta w(t_i)]^2$.

III. The periodic end conditions have

$$A = \begin{bmatrix} (\eta_{n-1} + \eta_1) & \gamma_1 & 0 & \cdots & 0 & \gamma_{n-1} \\ \gamma_1 & (\eta_1 + \eta_2) & \gamma_2 & & & 0 \\ 0 & \gamma_2 & & & & \vdots \\ \vdots & & & \ddots & & 0 \\ & & \ddots & & & \gamma_{n-2} \\ \gamma_{n-1} & 0 & \cdots & 0 & \gamma_{n-2} & (\eta_{n-2} + \eta_{n-1}) \end{bmatrix} \tag{15a}$$

$$c = (c_1, c_2, \ldots, c_{n-1})^T, \tag{15b}$$

$$g = 3(g_1^{(1)} - g_{n-1}^{(1)}, g_2^{(1)} - g_1^{(1)}, \ldots, g_{n-1}^{(1)} - g_{n-2}^{(1)}). \tag{15c}$$

The matrix of the system is tridiagonal except for the γ_{n-1} in the remote corners. This type of linear system is solved in DCSINT using a method described by Björck and Golub in [1], which uses a rank one modification to make (15a) tridiagonal. The number of multiplications and divisions needed to solve system (15) is about $9n$.

The linear systems arising for end conditions I, II, and III are all positive definite and symmetric with nonnegative coefficient matrices A. In all three cases the matrix is strictly diagonally dominant. The infinity condition number for the matrix A in (14a) and (15a) is bounded by (see [2])

$$\text{cond}_\infty(A) \le \|A\|_\infty \|A^{-1}\| \le 3\frac{\max_i(H_{i+1} + H_i)}{\min_i(H_{i+1} + H_i)}. \tag{16a}$$

For (13a) the bound is

$$\text{cond}_\infty(\text{A}) \le 3\frac{\max_i(H_{i+1} + H_i)}{\min_i(H_i)}. \tag{16b}$$

Hence the A matrices are normally very well conditioned.

Once the c_i's are known, the b_i and d_i in (1) are given by

$$d_i = \frac{1}{3H_i}[c_{i+1} - c_i], \tag{17}$$

$$b_i = g_i^{(1)} - \frac{H_i}{3}(2c_i + c_{i+1}) \tag{18}$$

for $i = 1, 2, \ldots, n-1$ (for the periodic case $c_n = c_1$).

In [5] Lyche gives error bounds for discrete cubic spline interpolation using (I) the first central divided difference end conditions. These bounds involve differences for $g(t)$. He also gives an $O(h^2)$ bound for the distance between the discrete cubic spline and the continuous cubic spline interpolating the same data.

3. Discrete Cubic Spline Smoothing

In [2] the author develops the theory for discrete natural spline smoothing. This type of smoothing was originally studied by Whittaker [8] in a slightly different form (see also [9, pp. 303–316]). The equations for the cubic case can be derived without too much difficulty by developing the cubic discrete analog of Theorem 14.1 given by Greville in [3]. This discrete analog is now stated.

THEOREM 3.1. *Let ρ, h, and W_i, $i = 1, 2, \ldots, n$, be positive real numbers. Then there exists a unique function $S(t)$ defined on $T_M = \{t_0, t_1, \ldots, t_M\}$ which minimizes*

$$\sigma(f(t)) = \rho \sum_{j=1}^{n} W_j(f(\tau_j) - g_j)^2 + \sum_{i=1}^{M-1} [\nabla\Delta f(t_i)]^2. \tag{19}$$

This $S(t)$ is the unique discrete natural cubic spline (see (1) or (6)) *satisfying*

$$S(\tau_i) = g_i - \frac{6h^3}{\rho W_i}[d_i - d_{i-1}] \tag{20}$$

where $d_i = (c_{i+1} - c_i)/(3H_i)$ and $d_0 = d_n = 0$.

The resulting system of linear equations for the c_i's in (1) or (6) is obtained from (14a-c) and (17) by putting $S_1^{(2)} = S_n^{(2)} = 0$ and replacing the g_i's used in computing the $g_i^{(1)}$'s by $\tilde{g}_i = S(\tau_i)$, where $S(\tau_i)$ is given by (21).

This linear system is five banded, positive definite, and symmetric (see [2]). The following equations describe this system. For $i = 2, 3, \ldots, n-1$ with the

understanding that $c_0 = c_1 = c_n = c_{n+1} = 0$,

$$\beta_{i-1}c_{i-2} + [\gamma_{i-1} - (\beta_{i-1} + \beta_i + \epsilon_{i-1})]c_{i-1} + [(\eta_{i-1} + \eta_i) + (2\beta_i + \epsilon_{i-1} + \epsilon_i)]c_i$$
$$+ [\gamma_i - (\beta_i + \beta_{i+1} + \epsilon_i)]c_{i+1} + \beta_{i+1}c_{i+2} = 3(g_i^{(1)} - g_{i-1}^{(1)}). \tag{21}$$

The γ_i, η_i, and $g_i(1)$ are given by (9)-(11),

$$\beta_i = \frac{6h^3}{\rho W_i H_{i-1} H_i} \tag{22}$$

for $i = 1, 2, \ldots, n$ with $H_0 = H_n = 1$ ($H_i = \tau_{i+1} - \tau_i$), and

$$\epsilon_i = \frac{(\beta_i H_{i-1} + \beta_{i+1} H_{i+1})}{H_i} \tag{23}$$

for $i = 1, 2, \ldots, n-1$.

The discrete cubic spline smoothing the data (τ_i, g_i) for $i = 1, 2, \ldots, n$ has the form $S(t) = S_i(t)$ for $t \in [\tau_i, \tau_{i+1}]$ with

$$S_i(t) = \tilde{g}_i + b_i(t - \tau_i) + c_i(t - \tau_i)^2 + d_i(t - \tau_i)^3 \tag{24}$$

where the c_i's solve (22), (17) gives d_i, (18) gives b_i, and $\tilde{g}_i = S(\tau_i)$ is given in (21).

The five-banded, positive definite, symmetric matrix arising from (22) need not be strictly diagonal dominant, as was the case for interpolation. The condition number of this matrix becomes large as ρ approaches 0. This is to be expected, since the solution to the smoothing problem is not unique when $\rho = 0$. From (23) and (24) we can see that as ρ becomes large, (22) takes the form of the equations for interpolation. Hence (22) is still reasonably conditioned for many useful values of ρ.

4. Examples for DCSINT

We now show how DCSINT is used to find the discrete cubic spline interpolating the data ($h = 0.1$, $n = 6$)

i	τ_i	g_i
1	0.5	1.0
2	0.7	0.5
3	1.0	2.0
4	1.5	2.5
5	2.1	2.0
6	2.5	1.0

for the three end conditions:

I. First central divided difference

$$\frac{1}{2}\frac{(\nabla + \Delta)S(\tau_1)}{h} = -1.0, \qquad \frac{1}{2}\frac{(\nabla + \Delta)S(\tau_n)}{h} = 0.0.$$

II. Second central divided difference

$$\frac{\nabla\Delta S(\tau_1)}{h^2} = 0.0, \qquad \frac{\nabla\Delta S(\tau_n)}{h^2} = 0.0.$$

(These are the natural end conditions.)

III. Periodic

$$S(\tau_1 - h) = S(\tau_n - h), \qquad S(\tau_1) = S(\tau_n), \qquad S(\tau_1 + h) = S(\tau_n + h).$$

A possible dimension statement for the calling program is

```
DIMENSION TNODE (10), G(10), B(10), C(10), D(10)
```

The nodes τ_i are stored in array TNODE. Note that TNODE (I) must be less than TNODE (I + 1). The data values g_i are stored in array G, and N = 6 and H = 0.1.

I. First Central Divided Difference End Conditions

Put IENT = 1, ENDI = −1.0, and ENDN = 0.0 and call the subroutine as follows:

CALL DCSINT (IENT, H, N, TNODE, G, ENDI, ENDN, B, C, D)

The solution is found in arrays G, B, C, and D.

Solution for I:

I	Interval	G(I)	B(I)	C(I)	D(I)
1	[0.5, 0.7]	1.0	−1.751037	−18.76556	75.10376
2	[0.7, 1.0]	0.5	0.9306393	2.629669	−42.44052
3	[1.0, 1.5]	2.0	4.736782	−11.89979	8.852451
4	[1.5, 2.1]	2.5	−0.4001307	1.378888	−3.501484
5	[2.1, 2.5]	2.0	−2.697525	−4.923788	13.54401

In particular, the discrete cubic spline $S(\tau)$ for $\tau = 1.2$ is given by

$$S(1.2) = 2.0 + 4.736782 \times T - 11.89979 \times T^2 + 8.852451 \times T^3 \qquad (25)$$

where $T = (1.2 - 1.0)$

II. Second Central Divided Difference End Conditions

Put IENT = 2, ENDI = 0.0, and ENDN = 0.0, and call DCSINT as before.

Solution for II:

I	G(I)	B(I)	C(I)	D(I)
1	1.0	−4.069061	0.0	39.22655
2	0.5	1.416796	23.53593	−38.63974
3	2.0	4.640036	−11.23984	7.9195391
4	2.5	−0.5627829	0.6394691	−1.817308
5	2.0	−1.798219	−2.631689	2.193076

III. Periodic End Conditions

Put IENT = 3. ENDI and ENDN may contain anything. Call DCSINT as before. (The subroutine assumes G(1) = G(N), so the content of G(N) is never used.)

Solution for III:

I	G(I)	B(I)	C(I)	D(I)
1	1.0	−3.801780	−2.156043	43.32474
2	0.5	1.357985	23.83881	−38.99586
3	2.0	4.663832	−11.25748	7.8596271
4	2.5	−0.6051248	0.5319628	−1.520514
5	2.0	−1.624539	−2.204966	0.04076882

5. An Example for DCSSMO

We now show how DCSSMO is used to find the discrete cubic natural spline which smoothes through the data used in Section 4. No end conditions are needed. A possible dimension statement for the calling program is

DIMENSION TNODE(10), G(10), GS(10), B(10), C(10), D(10), WGS(10)

The subroutine was run for ρ = RHO = 0.01 and for ρ = RHO = 1.0. The weights W_i, stored in array WGS, were taken to be $W_1 = W_5 = W_6 = 2.0$, $W_2 = 2.5$, $W_3 = 1.0$, and $W_4 = 1.5$. The subsolution is called as follows:

CALL DCSSMO (H, N, TNODE, G, WGS, RHO, GS, B, C, D)

The two solutions are as follows.

$\rho = 0.01$:

I	GS(I)	B(I)	C(I)	D(I)
1	0.7307289	1.536621	0.0	0.8975704
2	1.045234	1.667059	0.5385422	−1.374241
3	1.556716	1.611752	−0.6982756	−0.6354344
4	2.108594	0.4271141	−1.651427	0.3430809
5	1.844454	−1.189260	−1.033881	0.8615685

$\rho = 1.0$:

I	GS(I)	B(I)	C(I)	D(I)
1	0.9166568	−2.545056	0.0	27.78108
2	0.6298941	1.329906	16.66866	−26.34149
3	1.817824	3.915267	−7.038706	4.021049
4	2.518413	−0.06161821	−1.007132	−0.5822141
5	1.993115	−1.921916	−2.055118	1.712599

Theoretically, the discrete cubic spline smoothing the given data will approach, in the limit as ρ goes to infinity, the discrete natural cubic spline interpolating these data. From the graphs given in Figure 1 we can see that the discrete smoothing spline for $\rho = 1.0$ more closely follows the discrete natural interpolating spline than does the discrete smoothing spline for $\rho = 0.01$.

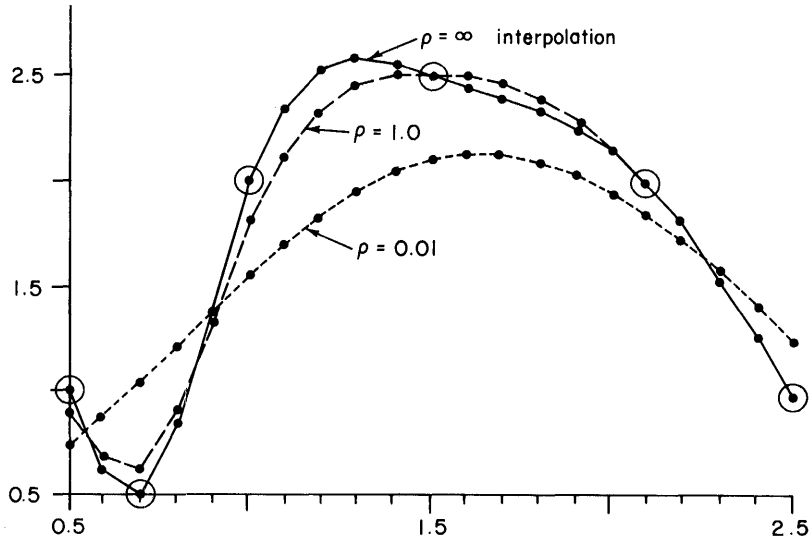


Fig. 1

6. Test Data

Any values may be used for test data g_i, $i = 1, 2, \ldots, n$, when interpolating. The correctness of the interpolating solution provided by DCSINT can be checked by verifying that the end conditions are satisfied and that

$$S_i(t) = S_{i+1}(t) \tag{26}$$

for $t = \tau_{i+1} - h$, τ_{i+1}, and $\tau_{i+1} + h$.

Test data for smoothing (DCSSMO) can be constructed from the discrete natural cubic spline represented in the form

$$S(t) = \alpha_0 + \alpha_1(t - \tau_1) + \sum_{i=1}^{n} \beta_i(t - \tau_i - h)(t - \tau_i)(t - \tau_i + h)\phi_i(t) \tag{27}$$

where

$$\phi_i(t) = \begin{cases} 0 & \text{for } t < \tau_i \\ 1 & \text{for } t \geq \tau_i \end{cases}$$

and the β_i's satisfy

$$\sum_{i=1}^{n} \beta_i = 0, \qquad \sum_{i=1}^{n} \beta_i \tau_i = 0. \tag{28}$$

The data values g_i which produce the $S(t)$ in (28) as the solution to the smoothing problem are given by

$$g_i = S(\tau_i) + 3! \frac{\beta_i}{(\rho W_i)} \tag{29}$$

for $i = 1, 2, \ldots, n$. This follows from Theorem 3.1, eq. (20).

7. Timing Estimates

DCSINT and DCSSMO have been run on the IBM 370/168 computer at the UNI-COLL Corporation in Philadelphia. The subroutines have been compiled and run using the WATFIV compiler and the Fortran G and H compilers. Timing estimates were made for the codes compiled by the G and H compilers. These are now listed.

DCSINT

For IENT = 1 (first central divided difference end conditions),

$$T_1 \approx 5.3N \times 10^{-5} \quad \text{seconds.}$$

For IENT = 2 (second central divided difference end conditions),

$$T_2 \approx 5.7N \times 10^{-5} \quad \text{seconds.}$$

For IENT = 3 (periodic end conditions),

$$T_3 \approx 7.3N \times 10^{-5} \quad \text{seconds.}$$

The above timing estimates are about right for the G compiler but are approximately 20 or 25 percent too large for the H compiler using the two-mode optimization parameter.

DCSSMO

$$T_S \approx 11N \times 10^{-5} \quad \text{seconds.}$$

Again this estimate is right for the G compiler but about 20 or 25 percent too large for the H compiler.

ACKNOWLEDGMENT

The author is indebted to two referees and to Prof. M.J.D. Powell for excellent comments and criticisms of an earlier version of this algorithm.

REFERENCES

1. BJÖRCK, A., AND GOLUB, G.H. Eigenproblems for matrices associated with periodic boundary conditions. *SIAM Rev. 19* (1977), 5–16.
2. DURIS, C.S. Discrete interpolation and smoothing spline functions. *SIAM J. Numer. Anal. 14* (1977), 686–698.
3. GREVILLE, T.N.E. Introduction to spline functions. In *Theory and Applications of Spline Functions*, T.N.E. Greville, Ed. Academic Press, New York, 1969, pp. 1–35.
4. HERRIOT, J.G., AND REINSCH, C.H. Algorithm 472: Procedures for natural spline interpolation. *Commun. ACM 16*, 12 (Dec. 1973), 763–768.
5. LYCHE, T. Discrete cubic spline interpolation. *BIT 16* (1976), 281–290.
6. LYCHE, T. Discrete cubic spline interpolation. Rep. RRI 5, University of Oslo, 1975.
7. SCHOENBERG, I.J. Spline functions and the problem of graduation. In Proc. Nat. Acad. Sci. (US) vol. 52, 1964, pp. 947–950.
8. WHITTAKER, E.T. On a new method of graduation. In Proc. Edinburgh Math. Soc., vol. 41, 1923, pp. 63–75.
9. WHITTAKER, E.T., AND ROBINSON, G. *The Calculus of Observations.* Blackie and Sons, Ltd., London, 1965.

ALGORITHM

```
      SUBROUTINE DCSSMO(H, N, TNODE, G, WGS, RHO, GSMO, B, C, D)        DCS   1Ø
C                                                                        DCS   2Ø
C  THIS SUBROUTINE COMPUTES THE DISCRETE NATURAL CUBIC                   DCS   3Ø
C  SPLINE DEFINED ON THE INTERVAL (TNODE(1),TNODE(N)) WHICH              DCS   4Ø
C  SMOOTHS THROUGH THE DATA (TNODE(I),G(I)),I=1,2,...,N.                 DCS   5Ø
C  N MUST BE 2 OR GREATER. THE NODES MUST SATISFY TNODE(I)               DCS   6Ø
C  .LT.TNODE(I+1). THE SOLUTION S(T) FOR T IN THE INTERVAL               DCS   7Ø
C  (TNODE(I),TNODE(I+1)) IS GIVEN BY                                     DCS   8Ø
```

```
C                                                                       DCS   9Ø
C     S(T)=GSMO(I)+B(I)*(T-TNODE(I))+                                   DCS  1ØØ
C             C(I)*(T-TNODE(I))**2+D(I)*(T-TNODE(I))**3                 DCS  11Ø
C                                                                       DCS  12Ø
      DIMENSION TNODE(N), G(N), WGS(N), GSMO(N), B(N), C(N), D(N)       DCS  13Ø
C                                                                       DCS  14Ø
C  INPUT  PARAMETERS(NONE OF THE INPUT PARAMETERS ARE CHANGED           DCS  15Ø
C         BY THIS SUBROUTINE)                                           DCS  16Ø
C                                                                       DCS  17Ø
C  H     - THE STEP SIZE USED FOR THE DISCRETE CUBIC SPLINE             DCS  18Ø
C  N     - NUMBER OF NODES (TNODE) AND DATA VALUES(G)                   DCS  19Ø
C  TNODE - REAL ARRAY CONTAINING THE NODES (TNODE(I).LT.                DCS  2ØØ
C          TNODE(I+1)).                                                 DCS  21Ø
C  G     - REAL ARRAY CONTAINING THE DATA VALUES.                       DCS  22Ø
C  WGS   - REAL ARRAY CONTAINING THE WEIGHTS WGS(I)                     DCS  23Ø
C          CORRESPONDING TO THE DATA (TNODE(I),G(I)).                   DCS  24Ø
C  RHO   - SIMPLE REAL VARIABLE CONTAINING THE POSITIVE                 DCS  25Ø
C          PARAMETER FOR VARYING THE SMOOTHNESS OF THE FIT.             DCS  26Ø
C          IF RHO IS SMALL SMOOTHNESS IS EMPHASIZED.                    DCS  27Ø
C          IF RHO IS LARGE DATA FITTING IS EMPHASIZED.                  DCS  28Ø
C                                                                       DCS  29Ø
C  OUTPUT PARAMETERS                                                    DCS  3ØØ
C                                                                       DCS  31Ø
C  GSMO  - REAL ARRAY CONTAINING THE SMOOTHED VALUES OF                 DCS  32Ø
C          THE DATA G(I),I=1,2,....,N.                                  DCS  33Ø
C  B     - REAL ARRAY CONTAINING THE COEFFICIENTS B(I) FOR               DCS  34Ø
C          THE TERMS (T-TNODE(I)).                                      DCS  35Ø
C  C     - REAL ARRAY CONTAINING THE COEFFICIENTS C(I) FOR               DCS  36Ø
C          THE TERMS (T-TNODE(I))**2.                                   DCS  37Ø
C  D     - REAL ARRAY CONTAINING THE COEFFICIENTS D(I) FOR               DCS  38Ø
C          THE TERMS (T-TNODE(I))**3.                                   DCS  39Ø
C                                                                       DCS  4ØØ
      IF (N.EQ.2) GO TO 18Ø                                             DCS  41Ø
      N1 = N - 1                                                        DCS  42Ø
      N2 = N1 - 1                                                       DCS  43Ø
      N3 = N2 - 1                                                       DCS  44Ø
C  THE RIGHT HAND SIDE OF THE LINEAR SYSTEM FOR THE                     DCS  45Ø
C  C(I)'S WILL NOW BE CONSTRUCTED.                                      DCS  46Ø
      DO 1Ø I=1,N                                                       DCS  47Ø
        C(I) = G(I)                                                     DCS  48Ø
   1Ø CONTINUE                                                          DCS  49Ø
      DO 2Ø I=1,N1                                                      DCS  5ØØ
        C(I) = (C(I+1)-C(I))/(TNODE(I+1)-TNODE(I))                      DCS  51Ø
   2Ø CONTINUE                                                          DCS  52Ø
      DO 3Ø I=1,N2                                                      DCS  53Ø
        C(I) = 3.Ø*(C(I+1)-C(I))                                        DCS  54Ø
   3Ø CONTINUE                                                          DCS  55Ø
C  THE RIGHT HAND SIDE IS NOW IN ARRAY C.                               DCS  56Ø
C                                                                       DCS  57Ø
C  THE P.D. 5 BANDED SYMMETRIC MATRIX WILL NOW BE CONSTRUCTED.          DCS  58Ø
C  THE THREE NEEDED DIAGONALS WILL BE STORED IN ARRAYS                  DCS  59Ø
C  GSMO,B,D.                                                            DCS  6ØØ
      H2 = H*H                                                          DCS  61Ø
      H3 = H2*H                                                         DCS  62Ø
      R6 = 6.Ø*H3/RHO                                                   DCS  63Ø
      HI3 = TNODE(2) - TNODE(1)                                         DCS  64Ø
      HI4 = TNODE(3) - TNODE(2)                                         DCS  65Ø
      ETA3 = HI3 + HI3 + H2/HI3                                         DCS  66Ø
      BETA3 = R6/(WGS(1)*HI3)                                           DCS  67Ø
      BETA4 = R6/(WGS(2)*HI3*HI4)                                       DCS  68Ø
      EPS3 = (BETA3+BETA4*HI4)/HI3                                      DCS  69Ø
      H2DHI = H2/HI4                                                    DCS  7ØØ
      ETA4 = HI4 + HI4 + H2DHI                                          DCS  71Ø
      IF (N.EQ.3) GO TO 6Ø                                              DCS  72Ø
      HI5 = TNODE(4) - TNODE(3)                                         DCS  73Ø
      BETA5 = R6/(WGS(3)*HI4*HI5)                                       DCS  74Ø
      EPS4 = (BETA4*HI3+BETA5*HI5)/HI4                                  DCS  75Ø
      GSMO(1) = ETA3 + ETA4 + BETA4 + BETA4 + EPS3 + EPS4               DCS  76Ø
      P = H2DHI + BETA4 + BETA5 + EPS4                                  DCS  77Ø
      B(1) = HI4 - P                                                    DCS  78Ø
      IF (N.EQ.4) GO TO 5Ø                                              DCS  79Ø
      DO 4Ø I=2,N3                                                      DCS  8ØØ
        HI3 = HI4                                                       DCS  81Ø
        HI4 = HI5                                                       DCS  82Ø
        HI5 = TNODE(I+3) - TNODE(I+2)                                   DCS  83Ø
        ETA3 = ETA4                                                     DCS  84Ø
```

```
            H2DHI = H2/HI4                                                    DCS  85Ø
            ETA4 = HI4 + HI4 + H2DHI                                          DCS  86Ø
            BETA3 = BETA4                                                     DCS  87Ø
            BETA4 = BETA5                                                     DCS  88Ø
            BETA5 = R6/(WGS(I+2)*HI4*HI5)                                     DCS  89Ø
            EPS3 = EPS4                                                       DCS  9ØØ
            EPS4 = (BETA4*HI3+BETA5*HI5)/HI4                                  DCS  91Ø
            D(I-1) = BETA4                                                    DCS  92Ø
            P = H2DHI + BETA4 + BETA5 + EPS4                                  DCS  93Ø
            B(I) = HI4 - P                                                    DCS  94Ø
            GSMO(I) = ETA3 + ETA4 + BETA4 + BETA4 + EPS3 + EPS4               DCS  95Ø
         4Ø CONTINUE                                                          DCS  96Ø
         5Ø HI3 = HI4                                                         DCS  97Ø
            HI4 = HI5                                                         DCS  98Ø
            ETA3 = ETA4                                                       DCS  99Ø
            ETA4 = HI4 + HI4 + H2/HI4                                         DCS 1ØØØ
            BETA4 = BETA5                                                     DCS 1Ø1Ø
            EPS3 = EPS4                                                       DCS 1Ø2Ø
         6Ø BETA5 = R6/(WGS(N)*HI4)                                           DCS 1Ø3Ø
            EPS4 = (BETA4*HI3+BETA5)/HI4                                      DCS 1Ø4Ø
            GSMO(N2) = ETA3 + ETA4 + BETA4 + BETA4 + EPS3 + EPS4              DCS 1Ø5Ø
      C  THE P.D. 5 BANDED SYMMETRIC MATRIX IS COMPLETE.                      DCS 1Ø6Ø
      C  THE SYSTEM OF LINEAR EQUATION WILL NOW BE SOLVED FOR THE             DCS 1Ø7Ø
      C  C(I)'S.                                                              DCS 1Ø8Ø
            IF (N.GT.3) GO TO 7Ø                                              DCS 1Ø9Ø
            C(1) = C(1)/GSMO(1)                                               DCS 11ØØ
            GO TO 15Ø                                                         DCS 111Ø
         7Ø IF (N.GT.4) GO TO 8Ø                                              DCS 112Ø
            C(1) = (C(1)*GSMO(2)-C(2)*B(1))/(GSMO(1)*GSMO(2)-B(1)**2)         DCS 113Ø
            C(2) = (C(2)-C(1)*B(1))/GSMO(2)                                   DCS 114Ø
            GO TO 15Ø                                                         DCS 115Ø
      C  THIS SOLVE THE 5 BANDED SYSTEM WHEN K=N-2.GT.3.                      DCS 116Ø
         8Ø K = N2                                                            DCS 117Ø
            K1 = K - 1                                                        DCS 118Ø
            K2 = K1 - 1                                                       DCS 119Ø
            K3 = K2 - 1                                                       DCS 12ØØ
      C  THE 5 BANDED MATRIX WILL NOW BE FACTORED.                            DCS 121Ø
            B(1) = B(1)/GSMO(1)                                               DCS 122Ø
            D(1) = D(1)/GSMO(1)                                               DCS 123Ø
            P = GSMO(1)*B(1)                                                  DCS 124Ø
            GSMO(2) = GSMO(2) - P*B(1)                                        DCS 125Ø
            B(2) = (B(2)-P*D(1))/GSMO(2)                                      DCS 126Ø
            IF (K.EQ.3) GO TO 11Ø                                             DCS 127Ø
            D(2) = D(2)/GSMO(2)                                               DCS 128Ø
            IF (K.EQ.4) GO TO 1ØØ                                             DCS 129Ø
            DO 9Ø I=3,K2                                                      DCS 13ØØ
              I1 = I - 1                                                      DCS 131Ø
              I2 = I1 - 1                                                     DCS 132Ø
              P = GSMO(I1)*B(I1)                                              DCS 133Ø
              GSMO(I) = GSMO(I) - GSMO(I2)*(D(I2)**2) - P*B(I1)               DCS 134Ø
              B(I) = (B(I)-P*D(I1))/GSMO(I)                                   DCS 135Ø
              D(I) = D(I)/GSMO(I)                                             DCS 136Ø
         9Ø CONTINUE                                                          DCS 137Ø
        1ØØ P = GSMO(K2)*B(K2)                                                DCS 138Ø
            GSMO(K1) = GSMO(K1) - GSMO(K3)*(D(K3)**2) - P*B(K2)               DCS 139Ø
            B(K1) = (B(K1)-P*D(K2))/GSMO(K1)                                  DCS 14ØØ
        11Ø GSMO(K) = GSMO(K) - GSMO(K2)*(D(K2)**2) - GSMO(K1)*(B(K1)**2)     DCS 141Ø
      C  FACTORIZATION COMPLETE.                                              DCS 142Ø
      C  CARRY OUT FORWARD  AND BACKWARD SUBSTITUTION.                        DCS 143Ø
            C(2) = C(2) - B(1)*C(1)                                           DCS 144Ø
            DO 12Ø I=3,K                                                      DCS 145Ø
              I1 = I - 1                                                      DCS 146Ø
              I2 = I - 2                                                      DCS 147Ø
              C(I) = C(I) - B(I1)*C(I1) - D(I2)*C(I2)                         DCS 148Ø
        12Ø CONTINUE                                                          DCS 149Ø
            DO 13Ø I=1,K                                                      DCS 15ØØ
              C(I) = C(I)/GSMO(I)                                             DCS 151Ø
        13Ø CONTINUE                                                          DCS 152Ø
            C(K1) = C(K1) - B(K1)*C(K)                                        DCS 153Ø
            DO 14Ø I=2,K1                                                     DCS 154Ø
              J = K - I                                                       DCS 155Ø
              C(J) = C(J) - B(J)*C(J+1) - D(J)*C(J+2)                         DCS 156Ø
        14Ø CONTINUE                                                          DCS 157Ø
      C  THE 5 BANDED SYSTEM HAS BEEN SOLVED.THE SOLUTION IS IN               DCS 158Ø
            DO 13Ø I=1,K                                                      DCS 15ØØ
              C(I) = C(I)/GSMO(I)                                             DCS 151Ø
```

```
  13Ø CONTINUE                                                          DCS 152Ø
      C(K1) = C(K1) - B(K1)*C(K)                                        DCS 153Ø
      DO 14Ø I=2,K1                                                     DCS 154Ø
        J = K - I                                                       DCS 155Ø
        C(J) = C(J) - B(J)*C(J+1) - D(J)*C(J+2)                         DCS 156Ø
  14Ø CONTINUE                                                          DCS 157Ø
C  THE 5 BANDED SYSTEM HAS BEEN SOLVED.THE SOLUTION IS IN               DCS 158Ø
C  ARRAY C. THE COEFFICIENTS GSMO, B, C, AND D WILL NOW BE              DCS 159Ø
C  SET UP.                                                              DCS 16ØØ
  15Ø C(N) = Ø.Ø                                                        DCS 161Ø
      D(N) = Ø.Ø                                                        DCS 162Ø
      C(N1) = C(N2)                                                     DCS 163Ø
      HK1 = TNODE(N) - TNODE(N1)                                        DCS 164Ø
      D(N1) = -C(N1)/(3.Ø*HK1)                                          DCS 165Ø
      GSMO(N) = G(N) + R6*D(N1)/WGS(N)                                  DCS 166Ø
      IF (N.EQ.3) GO TO 17Ø                                             DCS 167Ø
      DO 16Ø I=2,N2                                                     DCS 168Ø
        K = N - I                                                       DCS 169Ø
        K1 = K + 1                                                      DCS 17ØØ
        HK2 = HK1                                                       DCS 171Ø
        HK1 = TNODE(K1) - TNODE(K)                                      DCS 172Ø
        C(K) = C(K-1)                                                   DCS 173Ø
        D(K) = (C(K1)-C(K))/(3.Ø*HK1)                                   DCS 174Ø
        GSMO(K1) = G(K1) - R6*(D(K1)-D(K))/WGS(K1)                      DCS 175Ø
        B(K1) = (GSMO(K1+1)-GSMO(K1))/HK2 - HK2*(C(K1)+C(K1)+C(K1+1))/  DCS 176Ø
     *    3.Ø                                                          DCS 177Ø
  16Ø CONTINUE                                                          DCS 178Ø
  17Ø C(1) = Ø.Ø                                                        DCS 179Ø
      HK2 = HK1                                                         DCS 18ØØ
      HK1 = TNODE(2) - TNODE(1)                                         DCS 181Ø
      D(1) = (C(2)-C(1))/(3.Ø*HK1)                                      DCS 182Ø
      GSMO(2) = G(2) - R6*(D(2)-D(1))/WGS(2)                            DCS 183Ø
      GSMO(1) = G(1) - R6*D(1)/WGS(1)                                   DCS 184Ø
      B(2) = (GSMO(3)-GSMO(2))/HK2 - HK2*(C(2)+C(2)+C(3))/3.Ø           DCS 185Ø
      B(1) = (GSMO(2)-GSMO(1))/HK1 - HK1*(C(1)+C(1)+C(2))/3.Ø           DCS 186Ø
C  THE DISCRETE CUBIC SMOOTHING SPLINE IS NOW COMPLETE.                 DCS 187Ø
      RETURN                                                            DCS 188Ø
C  THE TRIVIAL CASE WHEN N=2 IS HANDLED HERE.                           DCS 189Ø
  18Ø GSMO(1) = G(1)                                                    DCS 19ØØ
      GSMO(2) = G(2)                                                    DCS 191Ø
      B(1) = (G(2)-G(1))/(TNODE(2)-TNODE(1))                            DCS 192Ø
      C(1) = Ø.Ø                                                        DCS 193Ø
      D(1) = Ø.Ø                                                        DCS 194Ø
      RETURN                                                            DCS 195Ø
      END                                                               DCS 196Ø


      SUBROUTINE DCSINT(IENT, H, N, TNODE, G, END1, ENDN, B, C, D)      DCS   1Ø
C                                                                       DCS   2Ø
C  THIS SUBROUTINE COMPUTES THE DISCRETE CUBIC SPLINE                   DCS   3Ø
C  DEFINED ON THE INTERVAL (TNODE(1),TNODE(N)),WHICH INTER-             DCS   4Ø
C  POLATES THE DATA (TNODE(I),G(I)),I=1,2,...,N. WE REQUIRE             DCS   5Ø
C  THAT TNODE(I).LT.TNODE(I+1). END1 AND ENDN CONTAIN THE               DCS   6Ø
C  VALUES OF THE END CONDITIONS BEING USED.                             DCS   7Ø
C                                                                       DCS   8Ø
C  IF IENT=1,THE FIRST CENTRAL DIVIDED DIFFERENCE END                   DCS   9Ø
C  CONDITIONS ARE BEING USED.                                           DCS  1ØØ
C                                                                       DCS  11Ø
C  IF IENT=2,THE SECOND CENTRAL DIVIDED DIFFERENCE END                  DCS  12Ø
C  CONDITIONS ARE BEING USED.                                           DCS  13Ø
C                                                                       DCS  14Ø
C  IF IENT=3,THE PERODIC END CONDITIONS ARE BEING USED.                 DCS  15Ø
C  FOR THIS CASE THE CONTENTS OF G(N), END1,AND ENDN ARE                DCS  16Ø
C  IGNORED.                                                             DCS  17Ø
C                                                                       DCS  18Ø
C  FOR ALL THREE END CONDITIONS N MUST BE GREATER THAN OR               DCS  19Ø
C  EQUAL TO 2.                                                          DCS  2ØØ
C                                                                       DCS  21Ø
C  THE DISCRETE CUBIC SPLINE IS REPRESENTED BY PIECEWISE                DCS  22Ø
C  CUBIC POLYNOMIALS. FOR T IN THE INTERNAL (TNODE(I),TNODE             DCS  23Ø
C  (I+1)) THE CUBIC SPLINE IS                                           DCS  24Ø
C                                                                       DCS  25Ø
C     S(T)=G(I)+B(I)*(T-TNODE(I))                                       DCS  26Ø
C                          +C(I)*(T-TNODE(I))**2                        DCS  27Ø
C                                  +D(I)*(T-TNODE(I))**3                DCS  28Ø
C                                                                       DCS  29Ø
```

```
      DIMENSION TNODE(N), G(N), B(N), C(N), D(N)                        DCS  3ØØ
C                                                                       DCS  31Ø
C  INPUT PARAMETERS (NONE OF THESE PARAMETERS                           DCS  32Ø
C                          ARE CHANGED BY THIS SUBROUTINE.)             DCS  33Ø
C                                                                       DCS  34Ø
C  IENT  - SPECIFIES END CONDITIONS WHICH ARE IN EFFECT.                DCS  35Ø
C  H     - THE STEP SIZE USED FOR THE DISCRETE CUBIC SPLINE.            DCS  36Ø
C  N     - NUMBER OF NODES (TNODE) AND DATA VALUES (G).                 DCS  37Ø
C          (N.GE.2)                                                     DCS  38Ø
C  TNODE - REAL ARRAY CONTAINING THE NODES (TNODE(I).LT.                DCS  39Ø
C          TNODE(I+1)).                                                 DCS  4ØØ
C  G     - REAL ARRAY CONTAINING THE INTERPOLATING DATA.                DCS  41Ø
C  END1  - END CONDITION VALUE AT TNODE(1).                             DCS  42Ø
C  ENDN  - END CONDITION VALUE AT TNODE(N).                             DCS  43Ø
C                                                                       DCS  44Ø
C  OUTPUT PARAMETERS                                                    DCS  45Ø
C                                                                       DCS  46Ø
C  B     - REAL ARRAY CONTAINING COEFFICIENTS OF                        DCS  47Ø
C          (T-TNODE(I)),I=1,2,....,N-1.                                 DCS  48Ø
C  C     - REAL ARRAY CONTAINING COEFFICIENTS OF                        DCS  49Ø
C          (T-TNODE(I))**2,I=1,2,...,N-1.                               DCS  5ØØ
C  D     - REAL ARRAY CONTAINING COEFFICIENTS OF                        DCS  51Ø
C          (T-TNODE(I))**3,I=1,2,...,N-1.                               DCS  52Ø
C                                                                       DCS  53Ø
C  SPECIAL CASES ARE ACCOUNTED FOR HERE.                                DCS  54Ø
      IF (N.EQ.2 .AND. IENT.EQ.3) GO TO 22Ø                             DCS  55Ø
      H2 = H*H                                                          DCS  56Ø
      N1 = N - 1                                                        DCS  57Ø
      IF (N.EQ.2 .AND. IENT.EQ.2) GO TO 18Ø                             DCS  58Ø
      N2 = N1 - 1                                                       DCS  59Ø
C  THE SYMMETRIC TRIDIAGONAL(OR NEAR TRIDIAGONAL) LINEAR               DCS  6ØØ
C  SYSTEM WILL NOW BE SET UP FOR THE APPROPRIATE END                    DCS  61Ø
C  CONDITIONS                                                           DCS  62Ø
      HI = TNODE(2) - TNODE(1)                                          DCS  63Ø
      H2DHI = H2/HI                                                     DCS  64Ø
      ETA2 = HI + HI + H2DHI                                            DCS  65Ø
      GO TO (1Ø, 4Ø, 6Ø), IENT                                          DCS  66Ø
C  IENT=1 - FIRST CENTRAL DIVIDED DIFFERENCE END CONDITONS             DCS  67Ø
C  SET UP.                                                             DCS  68Ø
   1Ø B(1) = ETA2                                                       DCS  69Ø
      D(1) = HI - H2DHI                                                 DCS  7ØØ
      G2 = (G(2)-G(1))/HI                                               DCS  71Ø
      C(1) = 3.Ø*(G2-END1)                                              DCS  72Ø
      IF (N.EQ.2) GO TO 3Ø                                              DCS  73Ø
      DO 2Ø I=2,N1                                                      DCS  74Ø
        ETA1 = ETA2                                                     DCS  75Ø
        G1 = G2                                                         DCS  76Ø
        HI = TNODE(I+1) - TNODE(I)                                      DCS  77Ø
        H2DHI = H2/HI                                                   DCS  78Ø
        ETA2 = HI + HI + H2DHI                                          DCS  79Ø
        B(I) = ETA1 + ETA2                                              DCS  8ØØ
        D(I) = HI - H2DHI                                               DCS  81Ø
        G2 = (G(I+1)-G(I))/HI                                           DCS  82Ø
        C(I) = 3.Ø*(G2-G1)                                              DCS  83Ø
   2Ø CONTINUE                                                          DCS  84Ø
   3Ø B(N) = ETA2                                                       DCS  85Ø
      C(N) = 3.Ø*(ENDN-G2)                                              DCS  86Ø
      L = N                                                             DCS  87Ø
C  SET UP FOR (1) FIRST CENTRAL DIVIDED DIFFERENCE END                 DCS  88Ø
C  CONDITING COMPLETE.THE LINEAR EQUATIONS WILL BE NOW                  DCS  89Ø
C  SOLVED.                                                             DCS  9ØØ
      GO TO 11Ø                                                         DCS  91Ø
C  IENT=2 - SECOND CENTRAL DIVIDED DIFFERENCE END CONDITIONS            DCS  92Ø
C  SET UP.                                                             DCS  93Ø
   4Ø GAMMA = HI - H2DHI                                                DCS  94Ø
      G2 = (G(2)-G(1))/HI                                               DCS  95Ø
      DO 5Ø I=1,N2                                                      DCS  96Ø
        ETA1 = ETA2                                                     DCS  97Ø
        G1 = G2                                                         DCS  98Ø
        HI = TNODE(I+2) - TNODE(I+1)                                    DCS  99Ø
        H2DHI = H2/HI                                                   DCS 1ØØØ
        ETA2 = HI + HI + H2DHI                                          DCS 1Ø1Ø
        B(I) = ETA1 + ETA2                                              DCS 1Ø2Ø
        D(I) = HI - H2DHI                                               DCS 1Ø3Ø
        G2 = (G(I+2)-G(I+1))/HI                                         DCS 1Ø4Ø
        C(I) = 3.Ø*(G2-G1)                                              DCS 1Ø5Ø
```

```
   5Ø CONTINUE                                                    DCS 1Ø6Ø
      C(1) = C(1) - GAMMA*END1/2.Ø                                DCS 1Ø7Ø
      HI = TNODE(N) - TNODE(N1)                                   DCS 1Ø8Ø
      GAMMA = HI - H2/HI                                          DCS 1Ø9Ø
      C(N2) = C(N2) - GAMMA*ENDN/2.Ø                              DCS 11ØØ
C  STEP UP FOR (2) SECOND CENTRAL DIVIDED DIFFERENCE              DCS 111Ø
C  END CONDITIONS COMPLETE. THE LINEAR EQUATIONS WILL NOW         DCS 112Ø
C  BE SOLVED.                                                     DCS 113Ø
      IF (N2.EQ.1) GO TO 1ØØ                                      DCS 114Ø
      L = N2                                                      DCS 115Ø
      GO TO 11Ø                                                   DCS 116Ø
C  IENT=3 - PERIODIC END CONDITIONS SET UP.                       DCS 117Ø
   6Ø B(1) = ETA2                                                 DCS 118Ø
      D(1) = HI - H2DHI                                           DCS 119Ø
      DO 7Ø I=2,N1                                                DCS 12ØØ
        ETA1 = ETA2                                               DCS 121Ø
        HI = TNODE(I+1) - TNODE(I)                                DCS 122Ø
        H2DHI = H2/HI                                             DCS 123Ø
        ETA2 = HI + HI + H2DHI                                    DCS 124Ø
        B(I) = ETA1 + ETA2                                        DCS 125Ø
        D(I) = HI - H2DHI                                         DCS 126Ø
        C(I) = Ø.Ø                                                DCS 127Ø
   7Ø CONTINUE                                                    DCS 128Ø
      CT = D(N1)                                                  DCS 129Ø
      C(1) = CT                                                   DCS 13ØØ
      C(N1) = CT                                                  DCS 131Ø
      B(1) = B(1) + ETA2 - CT                                     DCS 132Ø
      B(N1) = B(N1) - CT                                          DCS 133Ø
      L = N1                                                      DCS 134Ø
      ITRANS = 1                                                  DCS 135Ø
      GO TO 12Ø                                                   DCS 136Ø
   8Ø G1 = (G(N1)-G(N2))/(TNODE(N1)-TNODE(N2))                    DCS 137Ø
      G2 = (G(1)-G(N1))/(TNODE(N)-TNODE(N1))                      DCS 138Ø
      DEN = (1.Ø+C(1)+C(N1))                                      DCS 139Ø
      CT = 3.Ø*(G2-G1)                                            DCS 14ØØ
      BS1 = CT*C(N1)                                              DCS 141Ø
      C(N1) = CT                                                  DCS 142Ø
      DO 9Ø I=1,N2                                                DCS 143Ø
        HI = TNODE(I+1) - TNODE(I)                                DCS 144Ø
        G1 = G2                                                   DCS 145Ø
        G2 = (G(I+1)-G(I))/HI                                     DCS 146Ø
        CT = 3.Ø*(G2-G1)                                          DCS 147Ø
        BS1 = BS1 + CT*C(I)                                       DCS 148Ø
        C(I) = CT                                                 DCS 149Ø
   9Ø CONTINUE                                                    DCS 15ØØ
      BS1 = BS1/DEN                                               DCS 151Ø
      C(1) = C(1) - BS1                                           DCS 152Ø
      C(N1) = C(N1) - BS1                                         DCS 153Ø
      ITRANS = Ø                                                  DCS 154Ø
      GO TO 14Ø                                                   DCS 155Ø
C  THE SET UP AND MOST OF THE DETAILS FOR SOLVING THE             DCS 156Ø
C  LINEAR EQUQTION FOR (3)  THE PERIODIC END CONDITIONS           DCS 157Ø
C  ARE COMPLETED.                                                 DCS 158Ø
C                                                                 DCS 159Ø
C  THE LINEAR EQUATION ARE SOLVED HERE.                           DCS 16ØØ
  1ØØ C(2) = C(1)/B(1)                                            DCS 161Ø
      GO TO 18Ø                                                   DCS 162Ø
  11Ø ITRANS = Ø                                                  DCS 163Ø
  12Ø L1 = L - 1                                                  DCS 164Ø
      DO 13Ø I=1,L1                                               DCS 165Ø
        T = D(I)/B(I)                                             DCS 166Ø
        B(I+1) = B(I+1) - T*D(I)                                  DCS 167Ø
        D(I) = T                                                  DCS 168Ø
  13Ø CONTINUE                                                    DCS 169Ø
C  THE LINEAR EQUATION SOLVER IS ENTERED AT THIS POINT            DCS 17ØØ
C  IF THE LU FACTORIZATION HAS ALREADY BEEN DONE.                 DCS 171Ø
  14Ø DO 15Ø I=1,L1                                               DCS 172Ø
        C(I+1) = C(I+1) - D(I)*C(I)                               DCS 173Ø
  15Ø CONTINUE                                                    DCS 174Ø
      C(L) = C(L)/B(L)                                            DCS 175Ø
      DO 16Ø I=1,L1                                               DCS 176Ø
        LI = L - I                                                DCS 177Ø
        C(LI) = C(LI)/B(LI) - D(LI)*C(LI+1)                       DCS 178Ø
  16Ø CONTINUE                                                    DCS 179Ø
      IF (ITRANS.GE.1) GO TO 8Ø                                   DCS 18ØØ
C  THE LINEAR SYSTEM HAS BEEN SOLVE FOR THE C-VECTOR              DCS 181Ø
```

```
      IF (IENT.EQ.3) C(N) = C(1)                                        DCS 182Ø
      IF (IENT.NE.2) GO TO 19Ø                                          DCS 183Ø
      DO 17Ø I=1,N2                                                     DCS 184Ø
        LI = N - I                                                      DCS 185Ø
        C(LI) = C(LI-1)                                                 DCS 186Ø
  17Ø CONTINUE                                                          DCS 187Ø
  18Ø C(1) = END1/2.Ø                                                   DCS 188Ø
      C(N) = ENDN/2.Ø                                                   DCS 189Ø
  19Ø C1 = C(1)                                                         DCS 19ØØ
      C2 = C(2)                                                         DCS 191Ø
      HI = TNODE(2) - TNODE(1)                                          DCS 192Ø
      IF (N.EQ.2) GO TO 21Ø                                             DCS 193Ø
      DO 2ØØ I=1,N2                                                     DCS 194Ø
        B(I) = (G(I+1)-G(I))/HI - HI*(C1+C1+C2)/3.Ø                     DCS 195Ø
        D(I) = (C2-C1)/(3.Ø*HI)                                         DCS 196Ø
        HI = TNODE(I+2) - TNODE(I+1)                                    DCS 197Ø
        C1 = C2                                                         DCS 198Ø
        C2 = C(I+2)                                                     DCS 199Ø
  2ØØ CONTINUE                                                          DCS 2ØØØ
  21Ø GN = G(1)                                                         DCS 2Ø1Ø
      IF (IENT.NE.3) GN = G(N)                                          DCS 2Ø2Ø
      B(N1) = (GN-G(N1))/HI - HI*(C1+C1+C2)/3.Ø                         DCS 2Ø3Ø
      D(N1) = (C2-C1)/(3.Ø*HI)                                          DCS 2Ø4Ø
C  THE INTERPOLATING DISCRETE CUBIC SPLINE HAS BEEN                     DCS 2Ø5Ø
C  CONSTRUCTED.                                                         DCS 2Ø6Ø
      RETURN                                                            DCS 2Ø7Ø
C  THE FOLLOWING HANDLES THE TRIVIAL PERIODIC CASE                      DCS 2Ø8Ø
C  (IENT.EQ.3) WHEN N.EQ.2.                                             DCS 2Ø9Ø
  22Ø B(1) = Ø.Ø                                                        DCS 21ØØ
      C(1) = Ø.Ø                                                        DCS 211Ø
      D(1) = Ø.Ø                                                        DCS 212Ø
      RETURN                                                            DCS 213Ø
      END                                                               DCS 214Ø
```

```
C TEST FOR DURIS ALGORITHM, JAN 1979                                    DU ØØØ1Ø
C  DRIVER FOR DCSSMO                                                    DU ØØØ2Ø
C  THIS DRIVER USES SUBROUTINE DCSSMO TO COMPUTE THE                    DU ØØØ3Ø
C  DISCRETE NATURAL CUBIC SPLINE WHICH SMOOTHS THROUGH                  DU ØØØ4Ø
C  THE DATA (TNODE(I),G(I)),I=1,2,...,N AS SPECIFIED                    DU ØØØ5Ø
C  BY THE SMOOTHING PARAMETER RHO AND THE WEIGHTS WGS(I),               DU ØØØ6Ø
C  I=1,2,...,N.                                                         DU ØØØ7Ø
      DIMENSION TNODE(2Ø), G(2Ø), B(2Ø), C(2Ø), D(2Ø), GSMO(2Ø),        DU ØØØ8Ø
     *  WGS(2Ø)                                                         DU ØØØ9Ø
C  THE NEXT READ STATEMENT BRINGS IN THE TOTAL NUMBER                   DU ØØ1ØØ
C  OF DATA POINTS N, THE STEP SIZE H , AND THE PARAMETER RHO            DU ØØ11Ø
C  SPECIFYING THE AMOUNT OF SMOOTHING DESIRED.                          DU ØØ12Ø
      READ (5,99999) N, H, RHO                                          DU ØØ13Ø
C  THE FOLLOW WRITE STATMENTS OUTPUT N,H,AND RHO AND                    DU ØØ14Ø
C  PREPARES TO OUTPUT DATA AND SOLUTION.                                DU ØØ15Ø
      WRITE (6,99998)                                                   DU ØØ16Ø
      WRITE (6,99997) N, H, RHO                                         DU ØØ17Ø
C  THE NEXT DO LOOP READS IN THE DATA VALUES                            DU ØØ18Ø
C  (TNODE(I),G(I)), AND THE WEIGHTS WGS(I) FOR                          DU ØØ19Ø
C  I=1,2,...,N AND AT THE SAME TIME WRITES THIS DATA                    DU ØØ2ØØ
C  AS OUTPUT TO THE USER.                                               DU ØØ21Ø
      DO 1Ø I=1,N                                                       DU ØØ22Ø
        READ (5,99995) TNODE(I), G(I), WGS(I)                           DU ØØ23Ø
        WRITE (6,99996) I, TNODE(I), G(I), WGS(I)                       DU ØØ24Ø
   1Ø CONTINUE                                                          DU ØØ25Ø
C  SUBROUTINE DCSMO PARAMETERS H, TNODE,G,WGS,                          DU ØØ26Ø
C   ARE AS SPECIFIED ABOVE.                                             DU ØØ27Ø
      DRHO=RHO                                                          DU ØØ28Ø
      NN=N                                                              DU ØØ29Ø
      DO 5Ø N=2,NN                                                      DU ØØ3ØØ
      DO 4Ø IRHO=1,3                                                    DU ØØ31Ø
      RHO=DRHO*1ØØ.**(IRHO-2)                                           DU ØØ32Ø
      WRITE (6,99998)                                                   DU ØØ33Ø
      WRITE (6,99997) N, H, RHO                                         DU ØØ34Ø
      DO 15 I=1,N                                                       DU ØØ35Ø
        WRITE (6,99996) I, TNODE(I), G(I), WGS(I)                       DU ØØ36Ø
   15 CONTINUE                                                          DU ØØ37Ø
      CALL DCSSMO(H, N, TNODE, G, WGS, RHO, GSMO, B, C, D)              DU ØØ38Ø
C  OUTPUT FROM DCSSMO IS GSMO(I),B(I),C(I),AND D(I) FOR                 DU ØØ39Ø
C  I=1,2,...,N-1. IN PARTICULAR THE DISCRETE CUBIC SPLINE               DU ØØ4ØØ
C  ON THE INTERVAL (TNODE(1),TNODE(I+1)Ø IS DESCRIBED BY                DU ØØ41Ø
```

```
C  THE CUBIC POLYNOMIAL                                                  DU ØØ42Ø
C                                                                        DU ØØ43Ø
C       S(T)=GSMO(I)+B(I)*(T-TNODE(I))+                                  DU ØØ44Ø
C           C(I)*(T-TNODE(I))**2+D(I)*(T-TNODE(I))**3                    DU ØØ45Ø
C                                                                        DU ØØ46Ø
      N1 = N - 1                                                         DU ØØ47Ø
C  THE SOLUTION IS NOW PRINTED OUT BY THE FOLLOWING STATEMENTS           DU ØØ48Ø
      WRITE (6,99994)                                                    DU ØØ49Ø
      DO 2Ø I=1,N1                                                       DU ØØ5ØØ
        WRITE (6,99993) I, GSMO(I), B(I), C(I), D(I)                     DU ØØ51Ø
   2Ø CONTINUE                                                           DU ØØ52Ø
      WRITE(6,99992)                                                     DU ØØ53Ø
      DO 3Ø I=1,N1                                                       DU ØØ54Ø
      T=TNODE(I+1)-TNODE(I)                                              DU ØØ55Ø
      SL=GSMO(I)                                                         DU ØØ56Ø
      SR=GSMO(I)+(B(I)+(C(I)+D(I)*T)*T)*T                                DU ØØ57Ø
      FL=B(I)+D(I)*H*H                                                   DU ØØ58Ø
      FR=FL+(2.*C(I)+3.*D(I)*T)*T                                        DU ØØ59Ø
      DL=2.*C(I)                                                         DU ØØ6ØØ
      DR=DL+6.*D(I)*T                                                    DU ØØ61Ø
      WRITE(6,99993)I,SL,SR,FL,FR,DL,DR                                  DU ØØ62Ø
3Ø    CONTINUE                                                           DU ØØ63Ø
4Ø    CONTINUE                                                           DU ØØ64Ø
5Ø    CONTINUE                                                           DU ØØ65Ø
      CALL TEST2                                                         DU ØØ66Ø
      STOP                                                               DU ØØ67Ø
99999 FORMAT (I1Ø, 2E1Ø.2)                                               DU ØØ68Ø
99998 FORMAT (///5H DATA)                                                DU ØØ69Ø
99997 FORMAT (3H N=, I3, 2X, 2HH=, F1Ø.4, 3X, 4HRHO=, F1Ø.4  //5H   I ,  DU ØØ7ØØ
     *  3X, 8HTNODE(I), 9X, 4HG(I), 7X, 6HWGS(I))                        DU ØØ71Ø
99996 FORMAT (I3, 3E16.7)                                                DU ØØ72Ø
99995 FORMAT (3E1Ø.2)                                                    DU ØØ73Ø
99994 FORMAT (//4ØH DISCRETE NATURAL CUBIC SMOOTHING SPLINE//4H  I ,     DU ØØ74Ø
     *  2X, 7HGSMO(I), 12X, 4HB(I), 16X, 4HC(I), 16X, 4HD(I))            DU ØØ75Ø
99993 FORMAT (I3, 6E17.7)                                                DU ØØ76Ø
99992 FORMAT(3HØ I,14X,6HVALUES,18X,25HFIRST DIVIDED DIFFERENCES,        DU ØØ77Ø
     1 9X,26HSECOND DIVIDED DIFFERENCES)                                 DU ØØ78Ø
      END                                                                DU ØØ79Ø


      SUBROUTINE TEST2                                                   DU ØØ8ØØ
C  DRIVER FOR DCSINT                                                     DU ØØ81Ø
C  THIS DRIVER USES SUBROUTINE DCSINT TO COMPUTE THE                     DU ØØ82Ø
C  DISCRETE CUBIC SPLINE WHICH INTERPOLATES THE DATA                     DU ØØ83Ø
C  (TNODE(I),G(I)),I=1,2,...,N SUBJECT TO THE END CONDITIONS             DU ØØ84Ø
C  SPECIFIED BY IENT,END1,AND ENDN.                                      DU ØØ85Ø
C                                                                        DU ØØ86Ø
      DIMENSION TNODE(2Ø), G(2Ø), B(2Ø), C(2Ø), D(2Ø)                    DU ØØ87Ø
C                                                                        DU ØØ88Ø
C  THE NEXT READ STATEMENT BRINGS IN THE NUMBER OF DATA                  DU ØØ89Ø
C  POINTS N AND THE DISCRETE CUBIC SPLINE STEP SIZE H.                   DU ØØ9ØØ
C                                                                        DU ØØ91Ø
      READ (5,99999) N, H                                                DU ØØ92Ø
      WRITE (6,99998) N, H                                               DU ØØ93Ø
      WRITE (6,99997)                                                    DU ØØ94Ø
C  THE NEXT DO LOOP READS IN AND WRITES OUT THE DATA                     DU ØØ95Ø
C  VALUES (TNODE(I),G(I)),I=1,2...,N.                                    DU ØØ96Ø
      DO 1Ø I=1,N                                                        DU ØØ97Ø
        READ (5,99996) TNODE(I), G(I)                                    DU ØØ98Ø
        WRITE (6,99995) I, TNODE(I), G(I)                                DU ØØ99Ø
   1Ø CONTINUE                                                           DU Ø1ØØØ
C  THE END CONDITIONS ARE NOW READ IN. IENT SPECIFIES                    DU Ø1Ø1Ø
C  THE TYPE OF END CONDITION(SEE SUBROUTINE COMMENTS)                    DU Ø1Ø2Ø
C  AND END1 AND ENDN SPECIFY THE VALUES OF THE END CONDITIONS.           DU Ø1Ø3Ø
      READ (5,99994) IENT, END1, ENDN                                    DU Ø1Ø4Ø
C  SUBROUTINE DCSINT IS NOW CALLED. THE PARAMETERS                       DU Ø1Ø5Ø
C  H,TNODE,G,END1,AND ENDN ARE AS SPECIFIED ABOVE                        DU Ø1Ø6Ø
      NN=N                                                               DU Ø1Ø7Ø
      DO 5Ø N=2,NN                                                       DU Ø1Ø8Ø
      WRITE (6,99998) N, H                                               DU Ø1Ø9Ø
      WRITE (6,99997)                                                    DU Ø11ØØ
      DO 15 I=1,N                                                        DU Ø111Ø
        WRITE (6,99995) I, TNODE(I), G(I)                                DU Ø112Ø
   15 CONTINUE                                                           DU Ø113Ø
      DO 4Ø IENT=1,3                                                     DU Ø114Ø
C                                                                        DU Ø115Ø
```

```
      CALL DCSINT(IENT, H, N, TNODE, G, END1, ENDN, B, C, D)          DU Ø116Ø
C  THE END CONDITIONS IENT,END1,ENDN AND THE DISCRETE                 DU Ø117Ø
C  CUBIC SPLINE SOLUTION TO THE INTERPOLATIONPROBLEM                  DU Ø118Ø
C  ARE NOW OUTPUTED BELOW. IN PARTICULAR THE DISCRETE                 DU Ø119Ø
C  CUBIC SPLINE ON THE INTERVAL (TNODE(I),TNODE(I+1))                 DU Ø12ØØ
C  IS DESCRIBES BY THE CUBIC POLYNOMIAL                               DU Ø121Ø
C                                                                     DU Ø122Ø
C       S(T)=G(I)+B(I)*ARG+C(I)*ARG**2                                DU Ø123Ø
C                 +D(I)*ARG**3                                        DU Ø124Ø
C                                                                     DU Ø125Ø
C  WHERE ARG=(T-TNODE(I)).                                            DU Ø126Ø
C                                                                     DU Ø127Ø
      WRITE (6,99993) IENT, END1, ENDN                               DU Ø128Ø
      WRITE (6,99992)                                                DU Ø129Ø
      N1 = N - 1                                                     DU Ø13ØØ
      DO 2Ø I=1,N1                                                   DU Ø131Ø
        WRITE (6,99991) I, G(I), B(I), C(I), D(I)                    DU Ø132Ø
   2Ø CONTINUE                                                       DU Ø133Ø
      WRITE(6,9999Ø)                                                 DU Ø134Ø
      DO 3Ø I=1,N1                                                   DU Ø135Ø
      T=TNODE(I+1)-TNODE(I)                                          DU Ø136Ø
      SL=G(I)                                                        DU Ø137Ø
      SR=G(I)+(B(I)+(C(I)+D(I)*T)*T)*T                               DU Ø138Ø
      FL=B(I)+D(I)*H*H                                               DU Ø139Ø
      FR=FL+(2.*C(I)+3.*D(I)*T)*T                                    DU Ø14ØØ
      DL=2.*C(I)                                                     DU Ø141Ø
      DR=DL+6.*D(I)*T                                                DU Ø142Ø
      WRITE(6,99991)I,SL,SR,FL,FR,DL,DR                              DU Ø143Ø
3Ø    CONTINUE                                                       DU Ø144Ø
4Ø    CONTINUE                                                       DU Ø145Ø
5Ø    CONTINUE                                                       DU Ø146Ø
      RETURN                                                         DU Ø147Ø
99999 FORMAT (I1Ø, E1Ø.2)                                            DU Ø148Ø
99998 FORMAT (///5H DATA//3H N=, I3, 2X, 2HH=, F1Ø.5)                DU Ø149Ø
99997 FORMAT (2H I, 3X, 8HTNODE(I), 1ØX, 4HG(I))                     DU Ø15ØØ
99996 FORMAT (2E1Ø.2)                                                DU Ø151Ø
99995 FORMAT (I3, 3E16.7)                                            DU Ø152Ø
99994 FORMAT (I1Ø, 2E1Ø.2)                                           DU Ø153Ø
99993 FORMAT (///6H IENT=, I2, 2X, 5HEND1=, E16.7, 2X, 5HEND2=, E16.7) DU Ø154Ø
99992 FORMAT (/2H I, 3X, 4HG(I), 13X, 4HB(I), 13X, 4HC(I), 13X, 4HD(I)) DU Ø155Ø
99991 FORMAT (I3, 6E17.7)                                            DU Ø156Ø
9999Ø FORMAT(3HØ I,14X,6HVALUES,18X,25HFIRST DIVIDED DIFFERENCES,    DU Ø157Ø
     1 9X,26HSECOND DIVIDED DIFFERENCES)                             DU Ø158Ø
      END                                                            DU Ø159Ø


        7  Ø.1       1ØØ.Ø                                          DU Ø161Ø
 Ø.5        1.Ø      2.Ø                                            DU Ø162Ø
  Ø.7       Ø.5     2.5                                             DU Ø163Ø
  1.Ø      2.Ø      1.Ø                                             DU Ø164Ø
  1.5       2.5      1.5                                            DU Ø165Ø
  2.1      2.Ø        2.Ø                                          DU Ø166Ø
  2.5       1.Ø      2.Ø                                            DU Ø167Ø
  3.Ø       Ø.1      2.Ø                                            DU Ø168Ø
        6  Ø.1                                                      DU Ø169Ø
 Ø.5        1.Ø                                                    DU Ø17ØØ
  Ø.7       Ø.5                                                    DU Ø171Ø
  1.Ø      2.Ø                                                     DU Ø172Ø
  1.5       2.5                                                    DU Ø173Ø
  2.1      2.Ø                                                     DU Ø174Ø
  2.5       1.Ø                                                    DU Ø175Ø
        2  Ø.ØØ1    Ø.ØØ2                                          DU Ø176Ø
```

ALGORITHM 548
Solution of the Assignment Problem [H]

GIORGIO CARPANETO and PAOLO TOTH
University of Bologna, Italy

Key Words and Phrases: assignment problem, Hungarian algorithm
CR Categories: 5.39, 8.3
Language: Fortran

DESCRIPTION

Problem

The algorithm presented in this paper solves the assignment problem of the following form: given an $n \times n$ cost matrix $(a_{i,j})$, find a permutation (C_i) of integers $1, \ldots, n$ that minimizes

$$T = \sum_{i=1}^{n} a_{i,Ci}.$$

It is supposed, without loss of generality, that the elements of the cost matrix are nonnegative integers.

Algorithm

To give the reader a better understanding of the algorithm proposed below, we define:

C_j	as the row assigned to column j ($j = 1, \ldots, n$);
LC_j	as the label of column j; if $LC_j = 0$, column j is unlabeled ($j = 1, \ldots, n$);
LR_i	as the label of row i; if $LR_i = 0$, row i is unlabeled ($i = 1, \ldots, n$);
T	as the assignment cost;
P_i	as the set containing the columns corresponding to the unassigned zero elements of row i of the cost matrix ($i = 1, \ldots, n$);
RH	as the set containing the current not-completely-explored rows;
U	as the set containing the unassigned rows;
first (s)	as the first element of set s;
next (s)	as the element following the last considered element of set s;
last (s)	as the last element of set s.

Step 1. [Initialization]
Set $C_j = 0$, for $j = 1, \ldots, n$;
set $U = \emptyset$, $N = \{i \mid 1 \leq i \leq n\}$.
Step 2. [Reduction of the initial cost matrix]
Set $S_j = \min_{i \in N} [a_{i,j}]$, for all $j \in N$;

Received 5 August 1976 and 23 October 1978.

Author's address: Istituto di Automatica, Facoltà di Ingegneria, Universita Di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy.

set $Q_i = \min_{j \in N} [a_{i,j} - S_j]$, for all $i \in N$;
set $a'_{i,j} = a_{i,j} - S_j - Q_i$, for all $i \in N, j \in N$;
set $P_i = \{j \in N \mid a'_{i,j} = 0\}$, for all $i \in N$; set $T = \sum_{k \in N} (S_k + Q_k)$.

Step 3. [Choice of the initial solution]
Set $i = 1$.
a. Set $j = first\,\{k \in P_i \mid C_k = 0\}$; if j exists, go to Step 3d. Otherwise, set $j = first\,\{P_i\}$.
b. Set $m = first\,\{k \in P_{C_j} \mid C_k = 0\}$; if m exists, go to Step 3c. Otherwise, set $j = next\,\{P_i\}$; if j exists, repeat Step 3b; if not, set $U = U \cup \{i\}$, go to Step 3e.
c. Set $C_m = C_j$, $P_{C_j} = P_{C_j} \cup \{j\} - \{m\}$.
d. Set $C_j = i$, $P_i = P_i - \{j\}$.
e. If $i < n$, set $i = i + 1$, go to Step 3a.

Step 4. [Search for a new assignment]
If set U is empty, stop (the optimal assignment is given by vector (C_j), the minimum cost is given by T). Otherwise, set $RH = \varnothing$; set $LR_k = LC_k = 0$, for $k = 1, 2, \ldots, n$; set $r = first\,\{U\}$, $LR_r = -1$.
a. If set P_r is empty, go to Step 4c. Otherwise, if set P_r has at least two elements, set $RH = RH \cup \{r\}$; in any case, set $l = first\,\{P_r\}$.
b. If $LC_l = 0$, set $LC_l = r$, if $C_l = 0$, go to Step 6; if not, set $r = C_l$, $LR_r = l$, go to Step 4a. Otherwise ($LC_l \neq 0$), if $r \in RH$, go to Step 4d.
c. If set RH is empty, go to Step 5. Otherwise, set $r = first\,\{RH\}$.
d. Set $l = next\,\{P_r\}$, if $l = last\,\{P_r\}$, set $RH = RH - \{r\}$; in any case, go to Step 4b.

Step 5. [Reduction of the current cost matrix]
Set $SLR = \{i \in N \mid LR_i \neq 0\}$, $SUC = \{j \in N \mid LC_j = 0\}$.
Set $H = \min_{i \in SLR, j \in SUC} [a'_{i,j}]$.
For all $i \in SLR, j \in SUC$: set $a'_{i,j} = a'_{i,j} - H$, if $a'_{i,j} = 0$, set $P_i = P_i \cup \{j\}$, $RH = RH \cup \{i\}$;
For all $i \in N - SLR, j \in N - SUC$: if $j \in P_i$, set $P_i = P_i - \{j\}$; in any case set $a'_{i,j} = a'_{i,j} + H$.
Set $T = T + H$, $r = first\,\{RH\}$, go to Step 4d.

Step 6. [Assignment of a new row]
Set $C_l = r$, $P_r = P_r - \{l\}$; if $LR_r < 0$, set $U = U - \{r\}$, go to Step 4.
Otherwise, set $l = LR_r$, $P_r = P_r \cup \{l\}$, $r = LC_l$, repeat Step 6.

The efficiency of the algorithm is mainly due to the pointer technique utilized to locate the unexplored rows and the zero elements of the current cost matrix. It is worthwhile to note that, as far as storage requirement is concerned, for each set P_i ($i = 1, \ldots, n$) only the pointer to the first element needs to be stored; in fact, if column j is contained in set P_i, the corresponding element of the current cost matrix (i.e., $a'_{i,j}$) is zero and it is thus possible to replace this element by the pointer to the column following j in set P_i.

We obtained a further improvement on the original Hungarian algorithm by modifying the choice of the initial solution, as described in Step 3 of the proposed algorithm.

Program

Fortran IV subroutine ASSCT, based on the algorithm previously presented, is completely self-contained and communication to it is made solely through the parameter list. The subroutine is called by means of the statement:

CALL ASSCT (N, A, C, T).

All the parameters are integer and their meanings are the following:

Input: N = number of rows and columns of the cost matrix;
A = cost matrix.
Output: C = assignment vector;
T = minimum assignment cost.

After execution of subroutine ASSCT, the values of the elements of the cost matrix are changed. Vector C must be dimensioned at least at N; matrix A at least at (N, N + 1). As presently dimensioned, the size limitation for ASSCT is $N \leq 200$.

Table I. Cost Range 1–100

	A		B		C	
n	Average	Maximum	Average	Maximum	Average	Maximum
50	0.16	0.20	0.21	0.31	0.23	0.33
100	0.54	0.66	0.93	1.15	1.10	1.27
150	0.96	1.10	2.10	2.81	2.79	3.42
200	1.40	1.68	4.37	5.27	5.38	7.24

Table II. Cost Range 1–1000

	A		B		C	
n	Average	Maximum	Average	Maximum	Average	Maximum
50	0.41	0.51	0.56	0.78	0.60	0.89
100	2.15	2.61	2.97	3.92	3.34	4.24
150	4.55	6.28	6.60	8.37	7.24	8.42
200	6.70	8.01	10.36	12.26	11.83	13.15

Table III. Cost Range 1–10,000

	A		B		C	
n	Average	Maximum	Average	Maximum	Average	Maximum
50	0.48	0.61	0.66	0.81	0.72	0.96
100	3.71	4.60	5.78	7.49	6.80	9.55
150	11.74	15.57	18.72	21.96	21.35	23.94
200	19.85	25.40	32.69	43.52	42.63	48.16

Table IV. Sparse Matrices, Cost Range 1–100, $n = 200$

Number of coefficients	1500	2250	3000	3750	4500
AP-AB	0.97	1.12	1.48	1.61	1.68
PD-AAL	1.63	1.14	1.89	1.29	1.80
SUPERT-2	1.26	1.57	1.98	2.17	2.53
ASSCT	15.29	5.80	6.49	2.62	2.17

Computational Results

Subroutine ASSCT was tested in a CDC 6600 with over 1500 random problems of varying sizes. No breakdown in the method occurred.

To evaluate the efficiency of the proposed algorithm, the computing times of subroutine ASSCT were compared with those of the most efficient algorithms for solution of the assignment problem for dense matrices [1, 3].

Tables I, II, and III show the computing times corresponding to the algorithms:

A: Subroutine ASSCT.
B: Hungarian algorithm as presented in [5] and coded in Fortran IV by the authors.
C: Algorithm presented in [3] (the Fortran IV program, coded by Bourgeois and Lassalle, is taken from the CDC library of CERN (Geneva, Switzerland).

In Tables I, II, and III the values of the cost matrix were generated as uniformly random integers in the ranges, respectively, 1–100, 1–1000, and 1–10,000. All codes were run on a CDC 6600. For each cost range, each algorithm, and each value of n, 20 different problems were solved, and the average and maximum computing times, expressed in seconds, are given.

The tables show that subroutine ASSCT is always superior to the other codes, mainly for large values of n and for small ranges of costs. In addition, for all codes the computing times get worse with the increase in size of coefficients; in fact, when the size increases, the number of zero elements of the current cost matrix decreases.

In order to evaluate the effect of sparseness on the performance of the proposed algorithm, the same test problems considered by Barr, Glover, and Klingman in [2] were solved on the same machine (a CDC-6600). Subroutine ASSCT was compared with the most efficient codes for the solution of sparse assignment problems, viz., codes AP-AB, PD-AAL, and SUPERT-2 presented, respectively, in [2], [6], and [1]; the corresponding computing times, expressed in seconds, are given in Table IV.[1]

Table IV shows that the codes designed to solve sparse assignment problems are superior to subroutine ASSCT for very sparse matrices because this subroutine does not include any mechanism for taking advantage of sparsity. However, the trends of the computing times indicate that for fairly dense matrices the performance of the proposed algorithm greatly increases with respect to those of the other codes.

Further details of the algorithm and extensive computational results are given in [4].

REFERENCES

1. BARR, R.S. Streamling primal simplex transportation codes. Res. Rep., Center for Cybernetic Studies, U. of Texas, Austin, Tex. To appear.
2. BARR, R.S., GLOVER, F., AND KLINGMAN, D. The alternating basis algorithm for assignment problems. *Math. Programming 13* (1977), 1–13.
3. BOURGEOIS, F., AND LASSALLE, J.C. An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Comm. ACM 14*, 12 (Dec. 1971), 802–804.
4. CARPANETO, G., AND TOTH, P. An efficient algorithm for the assignment problem. Tech. Rep. 39, Istituto di Automatica, U. of Bologna, Bologna, Italy, 1976.
5. CHRISTOFIDES, N. *Graph Theory: An Algorithmic Approach.* Academic Press, London, 1975.
6. HATCH, R.S. Optimization strategies for large scale assignment and transportation type problems. ORSA/TIMS Conf., San Juan, Puerto Rico, 1974.

ALGORITHM

```
      SUBROUTINE ASSCT ( N, A, C, T )                                    1Ø
      INTEGER A(2ØØ,2Ø1), C(2ØØ), CH(2ØØ), LC(2ØØ), LR(2ØØ),             2Ø
     *        LZ(2ØØ), NZ(2ØØ), RH(2Ø1), SLC(2ØØ), SLR(2ØØ),             3Ø
     *        U(2Ø1)                                                     4Ø
      INTEGER H, Q, R, S, T                                              5Ø
      EQUIVALENCE (LZ,RH), (NZ,CH)                                       6Ø
C                                                                        7Ø
C THIS SUBROUTINE SOLVES THE SQUARE ASSIGNMENT PROBLEM                   8Ø
C THE MEANING OF THE INPUT PARAMETERS IS                                 9Ø
C N = NUMBER OF ROWS AND COLUMNS OF THE COST MATRIX, WITH                1ØØ
C     THE CURRENT DIMENSIONS THE MAXIMUM VALUE OF N IS 2ØØ               11Ø
C A(I,J) = ELEMENT IN ROW I AND COLUMN J OF THE COST MATRIX              12Ø
C ( AT THE END OF COMPUTATION THE ELEMENTS OF A ARE CHANGED)             13Ø
C THE MEANING OF THE OUTPUT PARAMETERS IS                                14Ø
C C(J) = ROW ASSIGNED TO COLUMN J (J=1,N)                                15Ø
C T = COST OF THE OPTIMAL ASSIGNMENT                                     16Ø
C ALL PARAMETERS ARE INTEGER                                             17Ø
C THE MEANING OF THE LOCAL VARIABLES IS                                  18Ø
C A(I,J) = ELEMENT OF THE COST MATRIX IF A(I,J) IS POSITIVE,             19Ø
C          COLUMN OF THE UNASSIGNED ZERO FOLLOWING IN ROW I              1J
C          (I=1,N) THE UNASSIGNED ZERO OF COLUMN J (J=1,N)               21Ø
C          IF A(I,J) IS NOT POSITIVE                                     22Ø
C A(I,N+1) = COLUMN OF THE FIRST UNASSIGNED ZERO OF ROW I                23Ø
C            (I=1,N)                                                     24Ø
C CH(I) = COLUMN OF THE NEXT UNEXPLORED AND UNASSIGNED ZERO              25Ø
C         OF ROW I (I=1,N)                                               26Ø
C LC(J) = LABEL OF COLUMN J (J=1,N)                                      27Ø
C LR(I) = LABEL OF ROW I (I=1,N)                                         28Ø
C LZ(I) = COLUMN OF THE LAST UNASSIGNED ZERO OF ROW I(I=1,N)             29Ø
C NZ(I) = COLUMN OF THE NEXT UNASSIGNED ZERO OF ROW I(I=1,N)             3ØØ
C RH(I) = UNEXPLORED ROW FOLLOWING THE UNEXPLORED ROW I                  31Ø
C         (I=1,N)                                                        32Ø
C RH(N+1) = FIRST UNEXPLORED ROW                                         33Ø
C SLC(K) = K-TH ELEMENT CONTAINED IN THE SET OF THE LABELLED             34Ø
C          COLUMNS                                                       35Ø
```

[1] The times in Table IV were communicated to the authors by an anonymous referee who has given us permission to publish them.

```
C SLR(K) = K-TH ELEMENT CONTAINED IN THE SET OF THE LABELLED              36Ø
C          ROWS                                                           37Ø
C U(I) = UNASSIGNED ROW FOLLOWING THE UNASSIGNED ROW I                    38Ø
C        (I=1,N)                                                          39Ø
C U(N+1) = FIRST UNASSIGNED ROW                                           4ØØ
C                                                                         41Ø
C THE VECTORS C,CH,LC,LR,LZ,NZ,SLC,SLR MUST BE DIMENSIONED                42Ø
C AT LEAST AT (N), THE VECTORS RH,U AT  LEAST AT (N+1),                   43Ø
C THE MATRIX A AT LEAST AT (N,N+1)                                        44Ø
C                                                                         45Ø
C INITIALIZATION                                                          46Ø
      MAXNUM = 1Ø**14                                                     47Ø
      NP1 = N+1                                                           48Ø
      DO 1Ø J=1,N                                                         49Ø
        C(J) = Ø                                                          5ØØ
        LZ(J) = Ø                                                         51Ø
        NZ(J) = Ø                                                         52Ø
        U(J) = Ø                                                          53Ø
   1Ø CONTINUE                                                            54Ø
      U(NP1) = Ø                                                          55Ø
      T = Ø                                                               56Ø
C REDUCTION OF THE INITIAL COST MATRIX                                    57Ø
      DO 4Ø J=1,N                                                         58Ø
        S = A(1,J)                                                        59Ø
        DO 2Ø L=2,N                                                       6ØØ
          IF ( A(L,J) .LT. S ) S = A(L,J)                                 61Ø
   2Ø   CONTINUE                                                          62Ø
        T = T+S                                                           63Ø
        DO 3Ø I=1,N                                                       64Ø
          A(I,J) = A(I,J)-S                                               65Ø
   3Ø   CONTINUE                                                          66Ø
   4Ø CONTINUE                                                            67Ø
      DO 7Ø I=1,N                                                         68Ø
        Q = A(I,1)                                                        69Ø
        DO 5Ø L=2,N                                                       7ØØ
          IF ( A(I,L) .LT. Q ) Q = A(I,L)                                 71Ø
   5Ø   CONTINUE                                                          72Ø
        T = T+Q                                                           73Ø
        L = NP1                                                           74Ø
        DO 6Ø J=1,N                                                       75Ø
          A(I,J) = A(I,J)-Q                                               76Ø
          IF ( A(I,J) .NE. Ø ) GO TO 6Ø                                   77Ø
          A(I,L) = -J                                                     78Ø
          L = J                                                           79Ø
   6Ø   CONTINUE                                                          8ØØ
   7Ø CONTINUE                                                            81Ø
C CHOICE OF THE INITIAL SOLUTION                                          82Ø
      K = NP1                                                             83Ø
      DO 14Ø I=1,N                                                        84Ø
        LJ = NP1                                                          85Ø
        J = -A(I,NP1)                                                     86Ø
   8Ø   IF ( C(J) .EQ. Ø ) GO TO 13Ø                                      87Ø
        LJ = J                                                            88Ø
        J = -A(I,J)                                                       89Ø
        IF ( J .NE. Ø ) GO TO 8Ø                                          9ØØ
        LJ = NP1                                                          91Ø
        J = -A(I,NP1)                                                     92Ø
   9Ø   R = C(J)                                                          93Ø
        LM = LZ(R)                                                        94Ø
        M = NZ(R)                                                         95Ø
  1ØØ   IF ( M .EQ. Ø ) GO TO 11Ø                                         96Ø
        IF ( C(M) .EQ. Ø ) GO TO 12Ø                                      97Ø
        LM = M                                                            98Ø
        M = -A(R,M)                                                       99Ø
      GO TO 1ØØ                                                          1ØØØ
  11Ø   LJ = J                                                           1Ø1Ø
        J = -A(I,J)                                                      1Ø2Ø
        IF ( J .NE. Ø ) GO TO 9Ø                                         1Ø3Ø
        U(K) = I                                                         1Ø4Ø
        K = I                                                            1Ø5Ø
        GO TO 14Ø                                                        1Ø6Ø
  12Ø   NZ(R) = -A(R,M)                                                  1Ø7Ø
        LZ(R) = J                                                        1Ø8Ø
        A(R,LM) = -J                                                     1Ø9Ø
        A(R,J) = A(R,M)                                                  11ØØ
        A(R,M) = Ø                                                       111Ø
        C(M) = R                                                         112Ø
```

```
  13Ø   C(J) = I                                                    113Ø
        A(I,LJ) = A(I,J)                                            114Ø
        NZ(I) = -A(I,J)                                             115Ø
        LZ(I) = LJ                                                  116Ø
        A(I,J) = Ø                                                  117Ø
  14Ø CONTINUE                                                      118Ø
C RESEARCH OF A NEW ASSIGNMENT                                      119Ø
  15Ø IF ( U(NP1) .EQ. Ø ) RETURN                                   12ØØ
      DO 16Ø I=1,N                                                  121Ø
        CH(I) = Ø                                                   122Ø
        LC(I) = Ø                                                   123Ø
        LR(I) = Ø                                                   124Ø
        RH(I) = Ø                                                   125Ø
  16Ø CONTINUE                                                      126Ø
      RH(NP1) = -1                                                  127Ø
      KSLC = Ø                                                      128Ø
      KSLR = 1                                                      129Ø
      R = U(NP1)                                                    13ØØ
      LR(R) = -1                                                    131Ø
      SLR(1) = R                                                    132Ø
      IF ( A(R,NP1) .EQ. Ø ) GO TO 22Ø                              133Ø
  17Ø L = -A(R,NP1)                                                 134Ø
      IF ( A(R,L) .EQ. Ø ) GO TO 18Ø                                135Ø
      IF ( RH(R) .NE. Ø ) GO TO 18Ø                                 136Ø
      RH(R) = RH(NP1)                                               137Ø
      CH(R) = -A(R,L)                                               138Ø
      RH(NP1) = R                                                   139Ø
  18Ø IF ( LC(L) .EQ. Ø ) GO TO 2ØØ                                 14ØØ
      IF ( RH(R) .EQ. Ø ) GO TO 21Ø                                 141Ø
  19Ø L = CH(R)                                                     142Ø
      CH(R) = -A(R,L)                                               143Ø
      IF ( A(R,L) .NE. Ø ) GO TO 18Ø                                144Ø
      RH(NP1) = RH(R)                                               145Ø
      RH(R) = Ø                                                     146Ø
      GO TO 18Ø                                                     147Ø
  2ØØ LC(L) = R                                                     148Ø
      IF ( C(L) .EQ. Ø ) GO TO 36Ø                                  149Ø
      KSLC = KSLC+1                                                 15ØØ
      SLC(KSLC) = L                                                 151Ø
      R = C(L)                                                      152Ø
      LR(R) = L                                                     153Ø
      KSLR = KSLR+1                                                 154Ø
      SLR(KSLR) = R                                                 155Ø
      IF ( A(R,NP1) .NE. Ø ) GO TO 17Ø                              156Ø
  21Ø CONTINUE                                                      157Ø
      IF ( RH(NP1) .GT. Ø ) GO TO 35Ø                               158Ø
C REDUCTION OF THE CURRENT COST MATRIX                              159Ø
  22Ø H = MAXNUM                                                    16ØØ
      DO 24Ø J=1,N                                                  161Ø
        IF ( LC(J) .NE. Ø ) GO TO 24Ø                               162Ø
        DO 23Ø K=1,KSLR                                             163Ø
          I = SLR(K)                                                164Ø
          IF ( A(I,J) .LT. H ) H = A(I,J)                           165Ø
  23Ø   CONTINUE                                                    166Ø
  24Ø CONTINUE                                                      167Ø
      T = T+H                                                       168Ø
      DO 29Ø J=1,N                                                  169Ø
        IF ( LC(J) .NE. Ø ) GO TO 29Ø                               17ØØ
        DO 28Ø K=1,KSLR                                             171Ø
          I = SLR(K)                                                172Ø
          A(I,J) = A(I,J)-H                                         173Ø
          IF ( A(I,J) .NE. Ø ) GO TO 28Ø                            174Ø
          IF ( RH(I) .NE. Ø ) GO TO 25Ø                             175Ø
          RH(I) = RH(NP1)                                           176Ø
          CH(I) = J                                                 177Ø
          RH(NP1) = I                                               178Ø
  25Ø     L = NP1                                                   179Ø
  26Ø     NL = -A(I,L)                                              18ØØ
          IF ( NL .EQ. Ø ) GO TO 27Ø                                181Ø
          L = NL                                                    182Ø
          GO TO 26Ø                                                 183Ø
  27Ø     A(I,L) = -J                                               184Ø
  28Ø   CONTINUE                                                    185Ø
  29Ø CONTINUE                                                      186Ø
      IF ( KSLC .EQ. Ø ) GO TO 35Ø                                  187Ø
      DO 34Ø I=1,N                                                  188Ø
        IF ( LR(I) .NE. Ø ) GO TO 34Ø                               189Ø
```

```
      DO 33Ø K=1,KSLC                                          19ØØ
        J = SLC(K)                                             191Ø
        IF ( A(I,J) .GT. Ø ) GO TO 32Ø                         192Ø
        L = NP1                                                193Ø
3ØØ     NL = - A(I,L)                                          194Ø
        IF ( NL .EQ. J ) GO TO 31Ø                             195Ø
        L = NL                                                 196Ø
        GO TO 3ØØ                                              197Ø
31Ø     A(I,L) = A(I,J)                                        198Ø
        A(I,J) = H                                             199Ø
        GO TO 33Ø                                              2ØØØ
32Ø     A(I,J) = A(I,J)+H                                      2Ø1Ø
33Ø   CONTINUE                                                 2Ø2Ø
34Ø CONTINUE                                                   2Ø3Ø
35Ø R = RH(NP1)                                                2Ø4Ø
    GO TO 19Ø                                                  2Ø5Ø
C ASSIGNMENT OF A NEW ROW                                      2Ø6Ø
36Ø C(L) = R                                                   2Ø7Ø
    M = NP1                                                    2Ø8Ø
37Ø NM = -A(R,M)                                               2Ø9Ø
    IF ( NM .EQ. L ) GO TO 38Ø                                 21ØØ
    M = NM                                                     211Ø
    GO TO 37Ø                                                  212Ø
38Ø A(R,M) = A(R,L)                                            213Ø
    A(R,L) = Ø                                                 214Ø
    IF ( LR(R) .LT. Ø ) GO TO 39Ø                              215Ø
    L = LR(R)                                                  216Ø
    A(R,L) = A(R,NP1)                                          217Ø
    A(R,NP1) = -L                                              218Ø
    R = LC(L)                                                  219Ø
    GO TO 36Ø                                                  22ØØ
39Ø U(NP1) = U(R)                                              221Ø
    U(R) = Ø                                                   222Ø
    GO TO 15Ø                                                  223Ø
    END                                                        224Ø
```

ALGORITHM 549
Weierstrass' Elliptic Functions [S21]

ULRICH ECKHARDT
University of Hamburg, Germany

Key Words and Phrases: Weierstrass' elliptic functions
CR Categories: 5.12
Language: Fortran

DESCRIPTION

Weierstrass' $\mathcal{P}$-function and its derivative can be approximated in the complex plane by a rational expression as described in [3, 4]. The following programs evaluate these approximations for the case of a unit period parallelogram. They are written according to the rules given in [2].

Programs PEQ and PEQ1 evaluate Weierstrass' $\mathcal{P}$-function in the equianharmonic case for a unit period parallelogram defined by the complex numbers

$$2\omega = \frac{1}{2} - \frac{1}{2}\sqrt{3}\,i, \qquad 2\omega' = \frac{1}{2} + \frac{1}{2}\sqrt{3}\,i$$

(see [1, Sec. 18.13] for notation). Because of the periodicity of Weierstrass' functions it is sufficient to calculate them only in the fundamental rectangle of all $z = x + iy$ with

$$-\frac{1}{2} \le x \le \frac{1}{2}, \qquad -\frac{\sqrt{3}}{4} \le y \le \frac{\sqrt{3}}{4}$$

which implies $|z| \le \sqrt{7}/4$. The theoretical absolute error of the approximation in this rectangle can be estimated as in [3] by the following bounds:

PEQ: 9.31×10^{-19}
PEQ1: 4.594×10^{-18}.

Numerical experiments were performed on different computers in order to obtain the actual errors of the programs. For large values of $|\mathcal{P}|$ the actual error is mainly due to the limited word length of the computer, whereas for small $|\mathcal{P}|$ the actual error approaches the theoretical error. Therefore a "mixed" error was calculated; it is the absolute error for $|\mathcal{P}| \le 1$ and the relative error elsewhere. The bounds for the mixed errors were obtained by comparing the programs with ϑ-series evaluations for randomly and systematically chosen arguments. Each program was tested on the computers given in Table I for more than 20,000 argument values.

Received 1 July 1977 and 19 March 1979.

Author's address: Institut für Angewandte Mathematik, Universität Hamburg, Bundesstrasse 55, 2 Hamburg 13, Germany.

Table I. Bounds for the Mixed Error for PEQ and PEQ1 on Different Computers

Computer	Mantissa length		PEQ	PEQ1
	Bits	Decimal		
IBM 370/168 (double precision)	56	16	2×10^{-15}	2×10^{-14}
CDC Cyber 172	48	14	7×10^{-13}	7×10^{-13}
TR 440	38	11	2×10^{-9}	3×10^{-9}

Table II. Bounds for the Mixed Error for PLEM and PLEM1

Computer	PLEM	PLEM1
IBM 370/168 (double precision)	4×10^{-15}	8×10^{-15}
CDC Cyber 172	6×10^{-13}	6×10^{-13}
TR 440	4×10^{-9}	5×10^{-9}
Theoretical absolute error	4.482×10^{-17}	8.033×10^{-17}

The difference between the numerical accuracy obtainable by using full word length of the computer and the values given in Table I is due to

(1) a loss of nearly one significant figure in computing powers of the variable z;
(2) a loss of one significant figure in evaluating the first line of the last assignment statement in each program.

Programs PLEM and PLEM1 evaluate Weierstrass' $\mathcal{P}$-function and its derivative in the lemniscatic case where

$$2\omega = 1, \qquad 2\omega' = i.$$

The fundamental rectangle is defined by

$$-\frac{1}{2} \le x \le \frac{1}{2}, \qquad -\frac{1}{2} \le y \le \frac{1}{2}$$

so that $|z| \le 1/\sqrt{2}$. The theoretical absolute error as given in [4] and the actual mixed error which was found by numerical experiments are given in Table II.

In using the programs it should be observed that the functions have poles at the lattice points $2\omega M + 2\omega' N$. If z happens to coincide with one of these poles, an overflow occurs. This can be avoided by checking after the reduction of z (see comments in the program listings) whether the reduced value of z is close to zero.

Full machine accuracy can be obtained in all programs by specifying the argument z and all its powers in greater precision and by performing all operations of the first line of the last assignment statement in double precision.

Note. The programs are written in single precision. On IBM computers they should be declared as

$$\text{COMPLEX FUNCTION} \left\{ \begin{array}{l} \text{PEQ} \\ \text{PEQ1} \\ \text{PLEM} \\ \text{PLEM1} \end{array} \right\} * 16(Z).$$

The first declaration should be COMPLEX * 16 and the second REAL * 8. All constants and functions should be given in double precision.

ACKNOWLEDGMENTS

The programs were written and tested on the IBM 370/168 of the Central Institute for Applied Mathematics of the Nuclear Research Center Jülich. They were also tested on the CDC Cyber 172 and the Telefunken TR 440 of the

Regionales Rechenzentrum Erlangen. The author is very much indebted to W. Alt of the University of Bayreuth for kind assistance.

REFERENCES

1. ABRAMOWITZ, M., AND STEGUN, I.A., Eds. Handbook of Mathematical Functions, with Formulas, Graphs, and Mathematical Tables. Nat. Bur. Standards (Appl. Math. Ser.), vol. 55. U.S. Government Printing Office, 1964. Reprint: Dover, New York, 1966.
2. CLARK, R.L. A linguistic contribution to GOTO-less programming. *Datamation 19*, (1973), 62–63.
3. ECKHARDT, U. A rational approximation to Weierstrass' $\mathcal{P}$-function. *Math. Comput. 30*, (1976), 818–826.
4. ECKHARDT, U. A rational approximation to Weierstrass' $\mathcal{P}$-function. II: The lemniscatic case. *Computing (Arch. elektron. Rechnen)* 18, (1977), 341–349.

ALGORITHM

```
      COMPLEX Z, PLEM, PLEM1, CR, CI, P, P1                              MAN   1Ø
      REAL W2, LC, ALPHA, FP, FP1, EPS, EPS1, A                          MAN   2Ø
C                                                                        MAN   3Ø
C     ***********************************************************************MAN   4Ø
C                                                                        MAN   5Ø
C     THE PROGRAMS FOR WEIERSTRASS: P-FUNCTION AND ITS DERIVATIVE IN THEMAN   6Ø
C     LEMNISCATIC CASE ARE TESTED FOR THE ARGUMENT VALUES                MAN   7Ø
C     Z = Ø.25, Ø.5, Ø.5 + Ø.5*I, Ø.25 + Ø.25*I                          MAN   8Ø
C     AND 1Ø.5 - 7.5*I                                                   MAN   9Ø
C     (SEE M. ABRAMOWITZ AND I. A. STEGUN& HANDBOOK OF MATHEMATICAL      MAN  1ØØ
C     FUNCTIONS, 18.14  FOR CORRESPONDING FUNCTION VALUES)               MAN  11Ø
C     THE DRIVER PROGRAM PRINTS THE MAXIMAL RELATIVE ERRORS OF P AND P:  MAN  12Ø
C                                                                        MAN  13Ø
C     ***********************************************************************MAN  14Ø
C                                                                        MAN  15Ø
C                                                                        MAN  16Ø
C     EVALUATION OF CONSTANTS                                            MAN  17Ø
C     =======================                                            MAN  18Ø
C                                                                        MAN  19Ø
      LC = 2.622Ø57554292119810Ø4648396                                  MAN  2ØØ
C                                                                        MAN  21Ø
C     LC IS THE LEMNISCATE CONSTANT                                      MAN  22Ø
C                                                                        MAN  23Ø
      W2 = 2.Ø*LC/SQRT(2.Ø)                                              MAN  24Ø
      ALPHA = 1.Ø + SQRT(2.Ø)                                            MAN  25Ø
      CR = (1.Ø,Ø.Ø)                                                     MAN  26Ø
      CI = (Ø.Ø,1.Ø)                                                     MAN  27Ø
      FP = 1.Ø/(W2*W2)                                                   MAN  28Ø
      FP1 = FP/W2                                                        MAN  29Ø
C                                                                        MAN  3ØØ
C     * * * * * * * * * *                                                MAN  31Ø
C                                                                        MAN  32Ø
      Z = (Ø.25,Ø.Ø)                                                     MAN  33Ø
      P = PLEM(Z)                                                        MAN  34Ø
      P1 = PLEM1(Z)                                                      MAN  35Ø
      EPS = CABS(P*FP-Ø.5*ALPHA*CR)/CABS(P)                              MAN  36Ø
      EPS1 = CABS(P1*FP1+ALPHA*CR)/CABS(P1)                              MAN  37Ø
C                                                                        MAN  38Ø
C     * * * * * * * * * *                                                MAN  39Ø
C                                                                        MAN  4ØØ
      Z = (Ø.5,Ø.Ø)                                                      MAN  41Ø
      P = PLEM(Z)                                                        MAN  42Ø
      P1 = PLEM1(Z)                                                      MAN  43Ø
      A = CABS(P*FP-Ø.5*CR)/CABS(P)                                      MAN  44Ø
      IF (A.GT.EPS) EPS = A                                              MAN  45Ø
      A = CABS(P1*FP1)                                                   MAN  46Ø
      IF (A.GT.EPS1) EPS1 = A                                            MAN  47Ø
C                                                                        MAN  48Ø
C     * * * * * * * * * *                                                MAN  49Ø
C                                                                        MAN  5ØØ
      Z = (Ø.5,Ø.5)                                                      MAN  51Ø
      P = PLEM(Z)                                                        MAN  52Ø
      P1 = PLEM1(Z)                                                      MAN  53Ø
      A = CABS(P*FP)                                                     MAN  54Ø
      IF (A.GT.EPS) EPS = A                                              MAN  55Ø
      A = CABS(P1*FP1)                                                   MAN  56Ø
      IF (A.GT.EPS1) EPS1 = A                                            MAN  57Ø
```

```
C                                                                       MAN  58Ø
C     * * * * * * * * * *                                               MAN  59Ø
C                                                                       MAN  6ØØ
      Z = (Ø.25,Ø.25)                                                   MAN  61Ø
      P = PLEM(Z)                                                       MAN  62Ø
      P1 = PLEM1(Z)                                                     MAN  63Ø
      A = CABS(P*FP+Ø.5*CI)/CABS(P)                                     MAN  64Ø
      IF (A.GT.EPS) EPS = A                                             MAN  65Ø
      A = CABS(P1*FP1-(CR+CI)/SQRT(2.Ø))/CABS(P1)                       MAN  66Ø
      IF (A.GT.EPS1) EPS1 = A                                           MAN  67Ø
C                                                                       MAN  68Ø
C     * * * * * * * * * *                                               MAN  69Ø
C                                                                       MAN  7ØØ
      Z = (1Ø.5,-7.5)                                                   MAN  71Ø
C                                                                       MAN  72Ø
C     THIS Z IS FOR TESTING THE REDUCTION TO FUNDAMENTAL PARALLELOGRAM. MAN  73Ø
C     BECAUSE OF THIS REDUCTION, THE ERROR MIGHT BE AN ORDER OF         MAN  74Ø
C     MAGNITUDE GREATER THAN THE BOUND GIVEN IN THE TEXT                MAN  75Ø
C                                                                       MAN  76Ø
      P = PLEM(Z)                                                       MAN  77Ø
      P1 = PLEM1(Z)                                                     MAN  78Ø
      A = CABS(P*FP)                                                    MAN  79Ø
      IF (A.GT.EPS) EPS = A                                             MAN  8ØØ
      A = CABS(P1*FP1)                                                  MAN  81Ø
      IF (A.GT.EPS1) EPS1 = A                                           MAN  82Ø
C                                                                       MAN  83Ø
C     * * * * * * * * * *                                               MAN  84Ø
C     * * * * * * * * * *                                               MAN  85Ø
C                                                                       MAN  86Ø
      WRITE (6,99999)                                                   MAN  87Ø
      WRITE (6,99998) EPS                                               MAN  88Ø
      WRITE (6,99997) EPS1                                              MAN  89Ø
      STOP                                                              MAN  9ØØ
99999 FORMAT (51H1   MAXIMAL RELATIVE ERROR FOR WEIERSTRASS P-FUNCTI,   MAN  91Ø
     * 2HON/49H    IN THE LEMNISCATIC CASE AT FIVE SAMPLE POINTS/4X,    MAN  92Ø
     * 5Ø(1H=))                                                         MAN  93Ø
99998 FORMAT ((///25H    RELATIVE ERROR FOR P=, 5X, E1Ø.3)              MAN  94Ø
99997 FORMAT ((//26H    RELATIVE ERROR FOR P1=, 4X, E1Ø.3)              MAN  95Ø
      END                                                               MAN  96Ø


      COMPLEX FUNCTION PEQ(Z)                                           PEQ   1Ø
C                                                                       PEQ   2Ø
C     WEIERSTRASS: P-FUNCTION IN THE EQUIANHARMONIC CASE                PEQ   3Ø
C     FOR COMPLEX ARGUMENT WITH UNIT PERIOD PARALLELOGRAM               PEQ   4Ø
C                                                                       PEQ   5Ø
      COMPLEX Z, Z2, Z4, Z6                                             PEQ   6Ø
      REAL ZR, ZI                                                       PEQ   7Ø
      INTEGER M, N                                                      PEQ   8Ø
C                                                                       PEQ   9Ø
C     REDUCTION TO FUNDAMENTAL PARALLELOGRAM                            PEQ  1ØØ
C                                                                       PEQ  11Ø
      ZI = 1.1547ØØ5383792515EØ*AIMAG(Z) + Ø.5EØ                        PEQ  12Ø
      M = INT(ZI)                                                       PEQ  13Ø
      IF (ZI.LT.ØEØ) M = M - 1                                          PEQ  14Ø
      ZR = REAL(Z) - Ø.5EØ*FLOAT(M) + Ø.5EØ                             PEQ  15Ø
      N = INT(ZR)                                                       PEQ  16Ø
      IF (ZR.LT.ØEØ) N = N - 1                                          PEQ  17Ø
      Z2 = Z - FLOAT(N) - (Ø.5EØ,Ø.866Ø254Ø378443865EØ)*FLOAT(M)        PEQ  18Ø
C                                                                       PEQ  19Ø
C     IF Z2=Ø THEN Z COINCIDES WITH A LATTICE POINT.                    PEQ  2ØØ
C     SINCE P HAS POLES AT THE LATTICE POINTS,                          PEQ  21Ø
C     A DIVISION ERROR WILL OCCUR                                       PEQ  22Ø
C                                                                       PEQ  23Ø
      Z2 = Z2*Z2                                                        PEQ  24Ø
      Z4 = Z2*Z2                                                        PEQ  25Ø
      Z6 = Z4*Z2                                                        PEQ  26Ø
      PEQ = 1EØ/Z2 + 6EØ*Z4*(5EØ+Z6)/(1EØ-Z6)**2 + Z4*                  PEQ  27Ø
     * (((((-2.6427662E-1Ø*Z6+1.61Ø954818E-8)*Z6+7.3861Ø752879E-6)*      PEQ  28Ø
     * Z6+4.3991444671178E-4)*Z6+7.4772882Ø49Ø697E-2)*                   PEQ  29Ø
     * Z6-6.848415328729921E-1)/(((((6.2252191E-1Ø*Z6+2.553314573E-7)*   PEQ  3ØØ
     * Z6-2.619832921421E-5)*Z6-5.64448Ø1847646E-4)*                     PEQ  31Ø
     * Z6+4.565553484821Ø6E-2)*Z6+1EØ)                                   PEQ  32Ø
      RETURN                                                            PEQ  33Ø
      END                                                               PEQ  34Ø
```

```
      COMPLEX FUNCTION PEQ1(Z)                                          PEQ   1Ø
C                                                                       PEQ   2Ø
C     FIRST DERIVATIVE OF WEIERSTRASS: P-FUNCTION IN THE                PEQ   3Ø
C     EQUIANHARMONIC CASE FOR COMPLEX ARGUMENT                          PEQ   4Ø
C     WITH UNIT PERIOD PARALLELOGRAM                                    PEQ   5Ø
C                                                                       PEQ   6Ø
      COMPLEX Z, Z3, Z6                                                 PEQ   7Ø
      REAL ZR, ZI                                                       PEQ   8Ø
      INTEGER M, N                                                      PEQ   9Ø
C                                                                       PEQ  1ØØ
C     REDUCTION TO FUNDAMENTAL PARALLELOGRAM                            PEQ  11Ø
C                                                                       PEQ  12Ø
      ZI = 1.1547ØØ5383792515EØ*AIMAG(Z) + Ø.5EØ                        PEQ  13Ø
      M = INT(ZI)                                                       PEQ  14Ø
      IF (ZI.LT.ØEØ) M = M - 1                                          PEQ  15Ø
      ZR = REAL(Z) - Ø.5EØ*FLOAT(M) + Ø.5EØ                             PEQ  16Ø
      N = INT(ZR)                                                       PEQ  17Ø
      IF (ZR.LT.ØEØ) N = N - 1                                          PEQ  18Ø
      Z3 = Z - FLOAT(N) - (Ø.5EØ,Ø.866Ø254Ø378443865EØ)*FLOAT(M)        PEQ  19Ø
C                                                                       PEQ  2ØØ
C     IF Z3=Ø THEN Z COINCIDES WITH A LATTICE POINT.                    PEQ  21Ø
C     SINCE P: HAS POLES AT THE LATTICE POINTS,                         PEQ  22Ø
C     A DIVISION ERROk WILL OCCUR                                       PEQ  23Ø
C                                                                       PEQ  24Ø
      Z3 = Z3*Z3*Z3                                                     PEQ  25Ø
      Z6 = Z3*Z3                                                        PEQ  26Ø
      PEQ1 = (((14EØ*Z6+294EØ)*Z6+126EØ)*Z6-2EØ)/(Z3*(1EØ-Z6)**3) +    PEQ  27Ø
     * Z3*((((((-2.95539175E-9*Z6-2.6764693Ø31E-7)*Z6+2.4Ø2192743346E-5)PEQ  28Ø
     * *Z6+1.9656661451391E-4)*Z6+1.76Ø135529461Ø36E-2)*                 PEQ  29Ø
     * Z6+8.1Ø26243498822636E-1)*Z6-2.7393661314919680 4EØ)/            PEQ  3ØØ
     * ((((((4.6397763E-1Ø*Z6+5.413482233E-8)*Z6-1.56293298374E-6)*     PEQ  31Ø
     * Z6-1.Ø3937Ø1Ø76352E-4)*Z6+9.5553182532237E-4)*                   PEQ  32Ø
     * Z6+9.1311Ø6969640212E-2)*Z6+1EØ)                                 PEQ  33Ø
      RETURN                                                            PEQ  34Ø
      END                                                               PEQ  35Ø


      COMPLEX FUNCTION PLEM(Z)                                          PLE   1Ø
C                                                                       PLE   2Ø
C     WEIERSTRASS: P-FUNCTION IN THE LEMNISCATIC CASE                   PLE   3Ø
C     FOR COMPLEX ARGUMENT WITH UNIT PERIOD PARALLELOGRAM               PLE   4Ø
C                                                                       PLE   5Ø
      COMPLEX Z, Z2, Z4, Z6                                             PLE   6Ø
      REAL ZR, ZI                                                       PLE   7Ø
      INTEGER M, N                                                      PLE   8Ø
C                                                                       PLE   9Ø
C     REDUCTION TO FUNDAMENTAL PARALLELOGRAM                            PLE  1ØØ
C                                                                       PLE  11Ø
      ZR = REAL(Z) + Ø.5EØ                                              PLE  12Ø
      ZI = AIMAG(Z) + Ø.5EØ                                             PLE  13Ø
      M = INT(ZR)                                                       PLE  14Ø
      N = INT(ZI)                                                       PLE  15Ø
      IF (ZR.LT.ØEØ) M = M - 1                                          PLE  16Ø
      IF (ZI.LT.ØEØ) N = N - 1                                          PLE  17Ø
      Z2 = Z - FLOAT(M) - (ØEØ,1EØ)*FLOAT(N)                            PLE  18Ø
C                                                                       PLE  19Ø
C     IF Z2=Ø THEN Z COINCIDES WITH A LATTICE POINT.                    PLE  2ØØ
C     SINCE P HAS POLES AT THE LATTICE POINTS,                          PLE  21Ø
C     A DIVISION ERROR WILL OCCUR                                       PLE  22Ø
C                                                                       PLE  23Ø
      Z2 = Z2*Z2                                                        PLE  24Ø
      Z4 = Z2*Z2                                                        PLE  25Ø
      Z6 = Z4*Z2                                                        PLE  26Ø
      PLEM = 1EØ/Z2 + 4EØ*Z2*(3EØ+Z4)/(1EØ-Z4)**2 +                     PLE  27Ø
     * Z2*(((((((-7.2331Ø8E-11*Z4+1.714197273E-8)*Z4-2.5369Ø36492E-7)* PLE  28Ø
     * Z4-7.9871Ø2Ø6868E-6)*Z4+6.485Ø6Ø69Ø9737E-4)*Z4+7.39624629362938E-PLE  29Ø
     * 3)*Z4+2.Ø12382768497244E-2)*Z4+7.1177297543136598E-1)*            PLE  3ØØ
     * Z4-2.5463639935383Ø738EØ)/(((((((5.1161516E-1Ø*Z4+6.61289Ø8E-9)PLE  31Ø
     * *Z4+4.4618987Ø48E-7)*Z4-8.42694918892E-6)*Z4+4.42886829Ø95E-6)*  PLE  32Ø
     * Z4-4.2262993521710 1E-3)*Z4+2.5774968717ØØ433E-2)*               PLE  33Ø
     * Z4+4.2359940482277Ø74E-1)*Z4+1EØ)                                PLE  34Ø
      RETURN                                                            PLE  35Ø
      END                                                               PLE  36Ø
```

```
      COMPLEX FUNCTION PLEM1(Z)                                          PLE   1Ø
C                                                                        PLE   2Ø
C     FIRST DERIVATIVE OF WEIERSTRASS: P-FUNCTION IN THE                 PLE   3Ø
C     LEMNISCATIC CASE FOR COMPLEX ARGUMENT                              PLE   4Ø
C     WITH UNIT PERIOD PARALLELOGRAM                                     PLE   5Ø
C                                                                        PLE   6Ø
      COMPLEX Z, Z1, Z3, Z4                                              PLE   7Ø
      REAL ZR, ZI                                                        PLE   8Ø
      INTEGER M, N                                                       PLE   9Ø
C                                                                        PLE  1ØØ
C     REDUCTION TO FUNDAMENTAL PARALLELOGRAM                             PLE  11Ø
C                                                                        PLE  12Ø
      ZR = REAL(Z) + Ø.5EØ                                               PLE  13Ø
      ZI = AIMAG(Z) + Ø.5EØ                                              PLE  14Ø
      M = INT(ZR)                                                        PLE  15Ø
      N = INT(ZI)                                                        PLE  16Ø
      IF (ZR.LT.ØEØ) M = M - 1                                           PLE  17Ø
      IF (ZI.LT.ØEØ) N = N - 1                                           PLE  18Ø
      Z1 = Z - FLOAT(M) - (ØEØ,1EØ)*FLOAT(N)                             PLE  19Ø
C                                                                        PLE  2ØØ
C     IF Z1=Ø THEN Z COINCIDES WITH A LATTICE POINT.                     PLE  21Ø
C     SINCE P: HAS POLES AT THE LATTICE POINTS,                          PLE  22Ø
C     A DIVISION ERROR WILL OCCUR                                        PLE  23Ø
C                                                                        PLE  24Ø
      Z3 = Z1*Z1*Z1                                                      PLE  25Ø
      Z4 = Z3*Z1                                                         PLE  26Ø
      PLEM1 = (((1E1*Z4+9E1)*Z4+3E1)*Z4-2EØ)/(Z1*(1EØ-Z4))**3 +          PLE  27Ø
     * Z1*((((((((((-3.9Ø463Ø2E-9*Z4-1.ØØ1487137E-8)*Z4+5.9573Ø43Ø92E-7)PLE  28Ø
     * *Z4-2.48251813Ø524E-5)*Z4+1.4557266595395E-4)*                   PLE  29Ø
     * Z4+4.5663365564326Ø6E-3)*Z4+6.224782572111135E-2)*                PLE  3ØØ
     * Z4+1.Ø38527937794269E-2)*Z4+1.198Ø462Ø8Ø2637942EØ)*               PLE  31Ø
     * Z4+6.427914396838117Ø8EØ)*Z4-5.Ø9272798Ø7661477EØ)/              PLE  32Ø
     * ((((((((((4.726888E-11*Z4-3.Ø667983E-9)*Z4+1.ØØ87596Ø89E-7)*     PLE  33Ø
     * Z4-8.Ø6Ø683451E-8)*Z4+1.184299251664E-5)*Z4-2.3Ø96723361547E-4)*  PLE  34Ø
     * Z4-2.9Ø73Ø9Ø3142Ø55E-3)*Z4+1.338392411135511E-2)*                 PLE  35Ø
     * Z4+2.3Ø9863932ØØ21426E-1)*Z4+8.4719880964554148E-1)*Z4+1EØ)       PLE  36Ø
      RETURN                                                             PLE  37Ø
      END                                                                PLE  38Ø


      COMPLEX Z, PEQ, PEQ1, CR, CI, P, P1                                MAN   1Ø
      REAL W2, SQ3, FP, FP1, EPS, EPS1, A, C                             MAN   2Ø
C                                                                        MAN   3Ø
C     ************************************************************************MAN   4Ø
C                                                                        MAN   5Ø
C     THE PROGRAMS FOR WEIERSTRASS: P-FUNCTION AND ITS DERIVATIVE IN THEMAN   6Ø
C     EQUIANHARMONIC CASE ARE TESTED FOR THE ARGUMENT VALUES             MAN   7Ø
C     Z = Ø.25, Ø.5, Ø.5 + Ø.5/SQRT(3)*I, Ø.25 + Ø.25/SQRT(3)*I          MAN   8Ø
C     AND 2.5 - 47/6*SQRT(3)*I                                           MAN   9Ø
C     (SEE M. ABRAMOWITZ AND I. A. STEGUN& HANDBOOK OF MATHEMATICAL      MAN  1ØØ
C     FUNCTIONS, 18.13. FOR CORRESPONDING FUNCTION VALUES)               MAN  11Ø
C     THE DRIVER PROGRAM PRINTS THE MAXIMAL RELATIVE ERRORS OF P AND P:  MAN  12Ø
C                                                                        MAN  13Ø
C     ************************************************************************MAN  14Ø
C                                                                        MAN  15Ø
C                                                                        MAN  16Ø
C     EVALUATION OF CONSTANTS                                            MAN  17Ø
C                                                                        MAN  18Ø
C     ======================                                             MAN  19Ø
C                                                                        MAN  2ØØ
      SQ3 = SQRT(3.Ø)                                                    MAN  21Ø
      CR = (1.Ø,Ø.Ø)                                                     MAN  22Ø
      CI = (Ø.Ø,1.Ø)                                                     MAN  23Ø
      C = 4.Ø**(-1.Ø/3.Ø)                                                MAN  24Ø
      W2 = 1.529954Ø37Ø5719287491319417231                               MAN  25Ø
      W2 = 2.Ø*W2                                                        MAN  26Ø
      FP = 1.Ø/(W2*W2)                                                   MAN  27Ø
      FP1 = FP/W2                                                        MAN  28Ø
C                                                                        MAN  29Ø
C     * * * * * * * * * *                                                MAN  3ØØ
C                                                                        MAN  31Ø
      Z = (Ø.25,Ø.Ø)                                                     MAN  32Ø
      P = PEQ(Z)                                                         MAN  33Ø
      P1 = PEQ1(Z)                                                       MAN  34Ø
      EPS = CABS(P*FP-C*(1.Ø+SQ3)*CR)/CABS(P)                            MAN  35Ø
      EPS1 = CABS(P1*FP1+3.Ø**Ø.75*SQRT(2.Ø+SQ3)*CR)/CABS(P1)            MAN  36Ø
```

```
C                                                                       MAN  37Ø
C     * * * * * * * * * *                                               MAN  38Ø
C                                                                       MAN  39Ø
      Z = (Ø.5,Ø.Ø)                                                     MAN  4ØØ
      P = PEQ(Z)                                                        MAN  41Ø
      P1 = PEQ1(Z)                                                      MAN  42Ø
      A = CABS(P*FP-C*CR)/CABS(P)                                       MAN  43Ø
      IF (A.GT.EPS) EPS = A                                             MAN  44Ø
      A = CABS(P1*FP1)                                                  MAN  45Ø
      IF (A.GT.EPS1) EPS1 = A                                           MAN  46Ø
C                                                                       MAN  47Ø
C     * * * * * * * * * *                                               MAN  48Ø
C                                                                       MAN  49Ø
      Z = Ø.5*CR + Ø.5/SQ3*CI                                           MAN  5ØØ
      P = PEQ(Z)                                                        MAN  51Ø
      P1 = PEQ1(Z)                                                      MAN  52Ø
      A = CABS(P*FP)                                                    MAN  53Ø
      IF (A.GT.EPS) EPS = A                                             MAN  54Ø
      A = CABS(P1*FP1-CI)/CABS(P1)                                      MAN  55Ø
      IF (A.GT.EPS1) EPS1 = A                                           MAN  56Ø
C                                                                       MAN  57Ø
C     * * * * * * * * * *                                               MAN  58Ø
C                                                                       MAN  59Ø
      Z = Z*Ø.5                                                         MAN  6ØØ
      P = PEQ(Z)                                                        MAN  61Ø
      P1 = PEQ1(Z)                                                      MAN  62Ø
      A = CABS(P*FP+2.Ø**(1.Ø/3.Ø)*Ø.5*(-CR+SQ3*CI))/CABS(P)            MAN  63Ø
      IF (A.GT.EPS) EPS = A                                             MAN  64Ø
      A = CABS(P1*FP1-(Ø.Ø,3.Ø))/CABS(P1)                               MAN  65Ø
      IF (A.GT.EPS1) EPS1 = A                                           MAN  66Ø
C                                                                       MAN  67Ø
C     * * * * * * * * * *                                               MAN  68Ø
C                                                                       MAN  69Ø
      Z = 2.5*CR - 47.Ø*SQ3/6.Ø*CI                                      MAN  7ØØ
C                                                                       MAN  71Ø
C     THIS Z IS FOR TESTING THE REDUCTION TO FUNDAMENTAL PARALLELOGRAM. MAN  72Ø
C     BECAUSE OF THIS REDUCTION, THE ERROR MIGHT BE AN ORDER OF         MAN  73Ø
C     MAGNITUDE GREATER THAN THE BOUND GIVEN IN THE TEXT                MAN  74Ø
C                                                                       MAN  75Ø
      P = PEQ(Z)                                                        MAN  76Ø
      P1 = PEQ1(Z)                                                      MAN  77Ø
      A = CABS(P*FP)                                                    MAN  78Ø
      IF (A.GT.EPS) EPS = A                                             MAN  79Ø
      A = CABS(P1*FP1-CI)/CABS(P1)                                      MAN  8ØØ
      IF (A.GT.EPS1) EPS1 = A                                           MAN  81Ø
C                                                                       MAN  82Ø
C     * * * * * * * * * *                                               MAN  83Ø
C     * * * * * * * * * *                                               MAN  84Ø
C                                                                       MAN  85Ø
      WRITE (6,99999)                                                   MAN  86Ø
      WRITE (6,99998) EPS                                               MAN  87Ø
      WRITE (6,99997) EPS1                                              MAN  88Ø
      STOP                                                              MAN  89Ø
99999 FORMAT (51H1   MAXIMAL RELATIVE ERROR FOR WEIERSTRASS P-FUNCTI,   MAN  9ØØ
     * 2HON/52H    IN THE EQUIANHARMONIC CASE AT FIVE SAMPLE POINTS/4X, MAN  91Ø
     * 5Ø(1H=))                                                         MAN  92Ø
99998 FORMAT (////25H    RELATIVE ERROR FOR P=, 5X, E1Ø.3)              MAN  93Ø
99997 FORMAT (//26H    RELATIVE ERROR FOR P1=, 4X, E1Ø.3)               MAN  94Ø
      END                                                               MAN  95Ø
```

ALGORITHM 550
Solid Polyhedron Measures [Z]

A.M. MESSNER
General Dynamics
and
G.Q. TAYLOR
I.T.T. Federal Electric Corporation

Key Words and Phrases: polyhedron, graphics, numerical integration
CR Categories: 3.2, 5.16, 8.2
Language: Fortran

DESCRIPTION

Surface area, centroid, volume, weight, moments, and products of inertia are computed by numerical integration over the bounding surfaces of solid polyhedra. This routine operates directly on geometric definitions consisting of vertex coordinates and face lists, in a format similar to the format employed in graphics and in plotting computer programs [1, 3, 5]. Integration is performed through a quadrature rule that is exact for the equations involved.

METHOD

The several equations, eqs. (1) through (10), to be integrated over the surface of the polyhedra were derived from volume integral forms through application of the Gauss divergence theorem.

$$\text{mass} = \frac{\rho}{3}\int\int [xn_x + yn_y + zn_z]\, dS = M \tag{1}$$

$$\text{CG}_x = \frac{\rho}{2M}\int\int [x^2 n_x]\, dS \tag{2}$$

$$\text{CG}_y = \frac{\rho}{2M}\int\int [y^2 n_y]\, dS \tag{3}$$

$$\text{CG}_z = \frac{\rho}{2M}\int\int [z^2 n_z]\, dS \tag{4}$$

$$I_{xx} = \frac{\rho}{3}\int\int [y^3 n_y + z^3 n_z]\, dS \tag{5}$$

Received 27 January 1976 and 6 November 1977.

Authors' addresses: A.M. Messner, General Dynamics Pomona Division, Pomona, CA; G.Q. Taylor, I.T.T. Federal Electric Corporation, RPL Project, Edwards Airforce Base, CA.

$$I_{yy} = \frac{\rho}{3} \iint [z^3 n_z + x^3 n_x]\, dS \tag{6}$$

$$I_{zz} = \frac{\rho}{3} \iint [x^3 n_x + y^3 n_y]\, dS \tag{7}$$

$$P_{xy} = \frac{-\rho}{2} \iint [y x^2 n_x]\, dS \tag{8}$$

$$P_{yz} = \frac{-\rho}{2} \iint [z y^2 n_y]\, dS \tag{9}$$

$$P_{zx} = \frac{-\rho}{2} \iint [x z^2 n_z]\, dS \tag{10}$$

where $\iint dS$ implies integration over the entire bounding surface; x, y, and z are right-hand Cartesian coordinates; n_x, n_y, and n_z are components of the outward directed normal vector of unit length; ρ is density per cubic unit; CG_x, CG_y, and CG_z are coordinates of the centroid; I_{xx}, I_{yy}, and I_{zz} are moments of inertia about the x, y, and z axes; P_{xy}, P_{yz}, and P_{zx} are products of inertia with respect to the subscripted axes with sign convention as shown.

Consider, for example, the moment of inertia about the x axis (as given in [2]):

$$I_{xx} = \iiint \rho(y^2 + z^2)\, dV.$$

We define a specific vector function $\mathbf{F}$ as follows:

$$\mathbf{F} = \frac{\rho}{3} \{y^3 \mathbf{j} + z^3 \mathbf{k}\}$$

where $\mathbf{j}$ and $\mathbf{k}$ are unit vectors in the coordinate directions y and z, respectively. The divergence of $\mathbf{F}$ is

$$\text{div } \mathbf{F} = \frac{\rho}{3} \left\{ \frac{\partial(y^3)}{\partial y} + \frac{\partial(z^3)}{\partial z} \right\} = \rho(y^2 + z^2).$$

Gauss's divergence theorem states

$$\iiint \text{div } \mathbf{F}\, dV = \iint \mathbf{F} \cdot \mathbf{n}\, dS$$

yielding

$$\iiint \rho(y^2 + z^2)\, dV = \iint \frac{\rho}{3} (y^3\, \mathbf{ny} + z^3\, \mathbf{nz})\, dS$$

which verifies that the surface integral form, eq. (5), is the equivalent of the volume representation.

Computation of the surface integrals is done through a quadrature rule applied to the triangular elements of the surface polygons. We employ a four-point rule, eq. (11), that is exact for polynomials up to degree three. This rule, a minimum-point exact rule for our purpose [6] converts the continuous integral into a sum of four discrete values evaluated at specific points in each triangle. Point locations and associated weights are as follows:

$$\iint f\, dA = \text{area}[f(p_1)w_1 + f(p_2)w_2 + f(p_3)w_3 + f(p_4)w_4] \tag{11}$$

where p_1 is the centroid of the triangle, $w_1 = -27/48$, $w_2 = w_3 = w_4 = 25/48$. p_2, p_3, and p_4 are located along lines from the centroid to the triangle vertices such that the distance to these points is equal to 40 percent of the distance from the centroid to the respective vertex.

Each surface polygon is treated as a group of triangles that are constructed as follows: Vertices 1, 2, and 3 form the first triangle, vertices 1, 3, and 4, form the second triangle, etc., with 1, $n - 1$, and n being the last triangle in a perimeter of n vertices. Normal vectors are computed for each triangle to ensure that concave polygons yield negative contributions when the perimeter reverses direction relative to the initial vertex. Contributions to mass, CG, products of inertia, and moments of inertia are accumulated in sequence to cascade the calculations and thereby reduce arithmetic operations. Global factors such as density are applied after all faces have been processed. Properties are finally shifted to parallel axes through the computed centroid. In this form, standard methods can be employed to yield values about arbitrary axes or to solve for principal values.

DESCRIPTION OF THE ALGORITHM

The algorithm is presented as two subroutines: PROPS and SRFINT, which are structured to operate on polyhedrons defined as a series of vertices and associated connectivity (faces) lists. Specific details of an appropriate data structure are illustrated by the example shown in Figure 1 and a driver program based on this format is included. Vertices are numbered, and defined in terms of their components in a right-hand rectangular Cartesian coordinate system. Faces are also numbered, and listed as a series of vertices in the order encountered while traversing the perimeters. Perimeters are traversed in a direction that is consistent with an inward surface vector definition. This ordering convention can be envisioned as follows: If one were to stand on the face with the left foot on the edge of the face and the right foot on the interior of the face and then traverse forward around the perimeter, vertices would be encountered in the required order. Perimeter lists are closed (the first vertex number is repeated at the end of the list) and each face consists of one or more perimeters. These aspects are illustrated by the example shown in Figure 1. Large faces are accommodated for in the driver program by continuing perimeter lists on subsequent cards using all data fields (including the first field normally used for the face number). A repeat of the initial vertex designates the end of the list.

Errors in the input data can occur, and are often difficult to detect unless systematic tests of data integrity are adopted. Geometric tests can be made to insure that the polyhedron is connected and has planar faces. These tests will vary with the application; nonplanar faces, for example, may be permitted in a mechanically deformed solid, relying on the triangulation process within the algorithm to enforce planarity. In our experience the most useful test has been a connectivity evaluation that sorts all edges and proceeds to match each edge against its reciprocal (same edge lists in the opposite order on another face). This test is easily accomplished while generating plots [3]. If each edge has one and only one reciprocal in the face lists there is reasonable assurance that the connectivity is valid. Another data error encountered in extensive polygon definitions is the omission of coordinates of one or more vertices. This error will usually result in coordinate values of zero, which can be tested, and appropriate cautionary messages printed out to aid the correction process. Large data sets may require graphic-generation techniques such as the use of digitizers or interactive entry devices. In most instances computer plots should be obtained as a further data check. Our experience is primarily with computations relating to designed components which are defined by engineering drawings. These drawings are reduced to appropriate card formats through digitizer equipment interfaced to a keypunch machine. Perspective plots (Figures 2 and 3) generated through a derivative of the Loutrel Algorithm [3] have been valuable guides to data integrity.

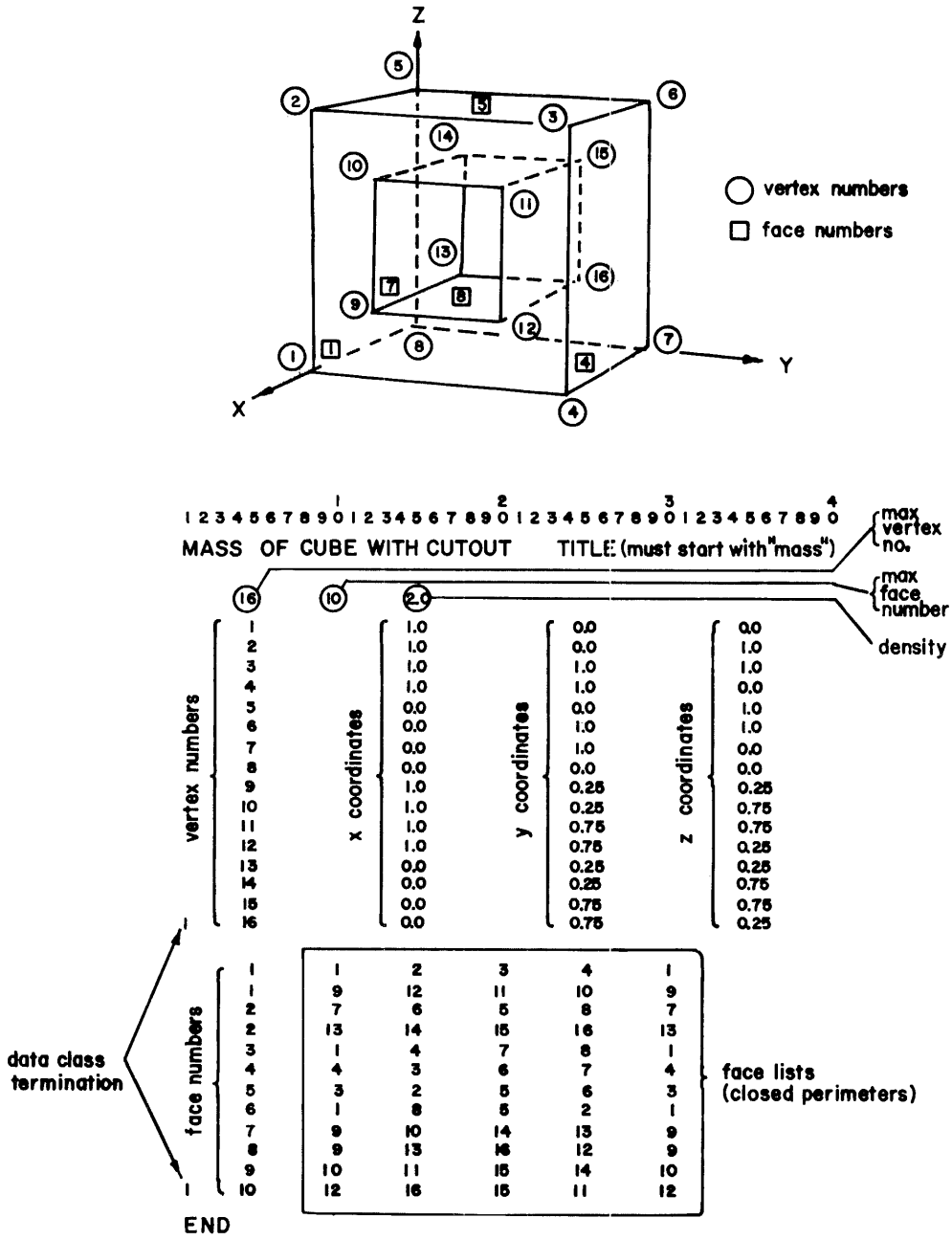


Fig. 1. Input definitions.

TESTS

This algorithm has been in use for several years and checked for simple geometries such as cubes and prisms as well as for complex configurations. Comparisons have also been made with fabricated components where computed values were compared with measured quantities of finished parts. Accuracies in all instances were within manufacturing tolerances. Computational errors arise from roundoff operations within the computer and can be minimized by multiple-precision arithmetic or through specific protection algorithms [4].

Several specific test problems are presented to illustrate the accuracies obtainable in single and double precision with the IBM 370 equipment. The first

Table I. Solutions Using Single and Double Precision with IBM 370 Computer

Single-Precision Solution to Test Problem 1 (100.0-inch cube with cutout 0.01-inch wall.)

Density	2.00000
Area	79999.8750
	(79999.9992)[a]
Volume	399.7500
	(399.96)[a]
Weight	799.50000
	(799.92)[a]

	Axes		
	x	*y*	*z*
CG			
coordinates	50.031	50.071	49.871
Moment	(50.0)[a]	(50.0)[a]	(50.0)[a]
from origin	6589440.00	5943488.00	5953024.00
from CG	2596525.00	1953777.00	1947323.00
	(2665866.666)[a]	(1999533.36)[a]	(1999533.36)[a]

	Axes		
	xy	*yz*	*zx*
Product moment			
from origin	1992960.00	1988096.00	1986816.00
from CG	−9891.000	−8345.000	−8029.000
	(0.0)[a]	(0.0)[a]	(0.0)[a]

Double-Precision Solution to Test Problem 1 (100.0-inch cube with cutout 0.01-inch wall.)

Density	2.00000
Area	79999.99920
Volume	399.95994
Weight	799.91988

	Axes		
	x	*y*	*z*
CG			
coordinates	50.00000	50.00000	50.00000
Moment			
from origin	6665464.73061	5999131.51563	5999131.51563
from CG	2665866.08647	1999532.87149	1999532.87149

	Axes		
	xy	*yz*	*zx*
Product moment			
from origin	1999799.32206	1999799.32206	1999799.32206
from CG	−0.000000	−0.00001	−0.00001

[a] True values shown below analogous computer results.

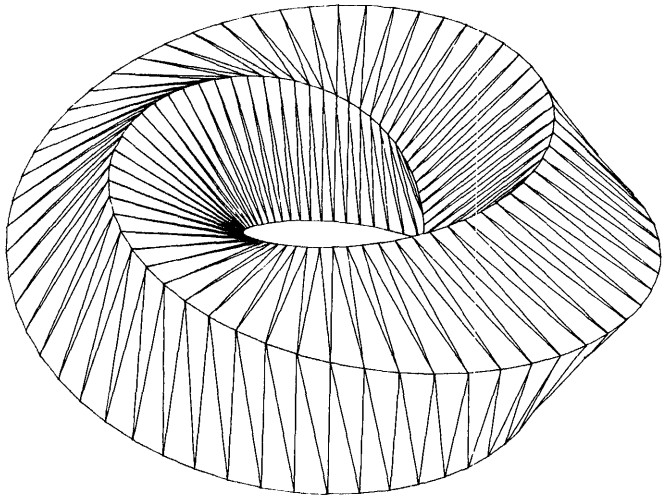


Fig. 2. Test Problem 2 generated solid.

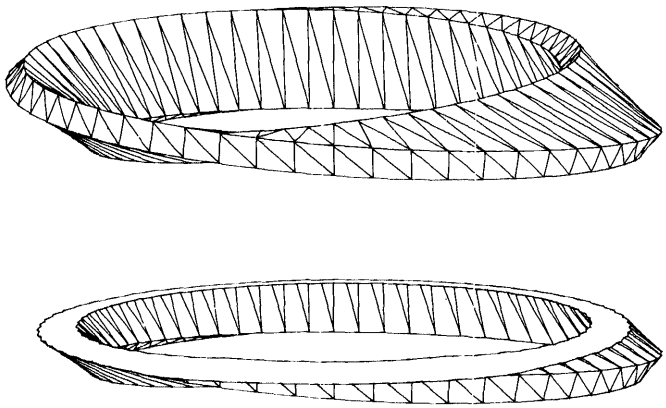


Fig. 3. Test Problem 3 Möbius and sectioned Möbius.

problem consists of a 100.0-inch cube, with cutout as in Figure 1. Cutout dimensions are such that the remaining wall thickness is 0.01 inch. Computations involve powers of coordinates which differ by the wall thickness; therefore, as the wall gets sufficiently thin, accuracy deteriorates. The complete single- and double-precision solution to this problem is presented in Table I. It can be seen that inaccuracies are evident in the single-precision solution, although even this problem, which is somewhat extreme for practical purposes, did not produce catastrophic errors. A second test has been constructed to indicate the accuracies related to geometries containing numerous faces with diverse orientations. This second test relates to the symmetry of results obtained for a generated configuration depicted in Figure 2. It was constructed by displacing an 8-inch square cross section normal to its plane in such a manner that its center traversed a 10-inch-radius circle while rotating 180 degrees in its plane during one traverse of the circle. The circle has its center at the origin and lies in the x–y plane. Single- and double-precision solutions are presented in Table II. Moments about x and y should be equal and the centroid and products of inertia components should be zero. It is of interest to note that the volume is some 2 percent smaller than would be obtained with a continuous solid generated in this manner. This discrepancy arises from the inscribing character of the polyhedron construction and would decrease as the number of faces increase. A final test is presented in Figure 3.

Table II. Computer Solutions to Example Problem 2

Single-Precision Solution to Problem 2			
Density	2.00000		
Area	2058.57715		
Volume	3941.82104		
Weight	7883.64062		
		Axes	
	x	*y*	*z*
CG			
coordinates	0.000	−0.000	0.000
Moment			
from origin	497116.062	497103.187	911423.062
from CG	497116.062	497103.187	911423.062
		Axes	
	xy	*yz*	*zx*
Product moment			
from origin	−23.484	−0.063	−0.033
from CG	−23.484	−0.063	−0.033
Double-Precision Solution to Problem 2			
Density	2.00000		
Area	2058.63206		
Volume	3942.03508		
Weight	7884.07016		
		Axes	
	x	*y*	*z*
CG			
coordinates	−0.00000	0.00000	0.00000
Moment			
from origin	497209.13864	497209.12703	911523.19783
from CG	497209.13864	497209.12703	911523.19783
		Axes	
	xy	*yz*	*zx*
Product moment			
from origin	−0.00155	0.00022	−0.00010
from CG	−0.00155	0.00022	−0.00010

Table III

Solution to Complete Solid Test Problem 3			
Density	2.00000		
Area	630.20361		
Volume	241.44048		
Weight	482.88086		
		Axes	
	x	*y*	*z*
CG			
coordinates	0.03247	--0.00031	0.00000
Moment			
from origin	24961.5117	24960.8477	49240.2383
from CG	24961.5117	24960.3359	49239.7266
		Axes	
	xy	*yz*	*zx*
Product moment			
from origin	−0.21539	−309.58984	0.00619
from CG	−0.21049	−309.58984	0.00618
Solution to Sectioned Solid Test Problem 3			
Density	2.00000		
Area	424.16016		
Volume	120.71504		
Weight	241.43008		
		Axes	
	x	*y*	*z*
CG			
coordinates	0.03247	--0.75071	0.67662
Moment			
from origin	12478.8516	12479.5312	24617.2148
from CG	12232.2578	12368.7422	24480.8984
		Axes	
	xy	*yz*	*zx*
Product moment			
from origin	−375.08545	−154.91779	−445.60889
from CG	−369.20068	−32.28427	−450.91260

This third test problem is similar in configuration to the second except that the cross section is now a 1- by 4-inch rectangle (finite thickness Möbius strip). A plane through the generating circle of this configuration bisects it identically. Solutions in single precision are presented to the complete and the sectioned solid in Table III where it can be seen that the volume, weight, moments about the axes, and the y-z product moment are halved as required. The x coordinate of the centroid is not zero due to the discretization errors associated with the geometry construction.

REFERENCES

1. GALIMBERTI, R., AND MONTANARI, U. An algorithm for hidden line elimination. *Comm. ACM 12*, 4 (April 1969), 206–211.
2. GOLDSTEIN, H. *Classical Mechanics.* Addison-Wesley, Reading, Mass., 1950, p. 145.
3. LOUTREL, P. A solution to the hidden line problem for computer drawn polyhedra. Thesis, New York University, New York, N.Y., 1968.
4. MALCOLM, M.A. On accurate floating point summation. *Comm. ACM 14*, 11 (Nov. 71), 731–736.
5. MESSNER, A., TAYLOR, G.Q., AND PRICE, C.F. Interdisciplinary computer analyses of three-dimensional solids defined by polyhedral surfaces. *J. Spacecraft and Rockets II* (Jan. 1974), 52–54.
6. STROUD, A.H. *Approximate Calculation of Multiple Integrals.* Prentice-Hall, Englewood Cliffs, N.J., 1971, p. 308.

ALGORITHM

```
C PURPOSE                                                              1Ø
C   TO CALCULATE MASS PROPERTIES OF GENERAL THREE                      2Ø
C   DIMENSIONAL SOLIDS.                                                3Ø
C                                                                      4Ø
C DESCRIPTION                                                          5Ø
C   THE SOLID IS DESCRIBED BY A SET OF SURFACES WHICH ARE              6Ø
C   TRIANGULATED, AND A FOUR POINT GAUSSIAN QUADRATURE OVER            7Ø
C   THE TRIANGLES IS USED IN CALCULATING SURFACE AREA,                 8Ø
C   VOLUME, AND MASS PROPERTIES.                                       9Ø
C                                                                     1ØØ
C                                                                     11Ø
      COMMON      MAXS,MAXV,                                          12Ø
     .            AREA,WEIGHT,VOL,DENS,                              13Ø
     .            CG(3),IO(3),ICG(3),PR(3),PRCG(3),                  14Ø
     .            ERROR,NERR,                                        15Ø
     .            HED(8Ø),IN,OT,KT,                                  16Ø
     .            N8,MAXKV                                           17Ø
      LOGICAL ERROR                                                   18Ø
      INTEGER OT                                                      19Ø
      INTEGER TITLE,END,HED                                           2ØØ
      DIMENSION A(1ØØØØ),KA(6ØØØ)                                     21Ø
      REAL IO,ICG                                                     22Ø
      DATA TITLE,END /1HM,1HE/                                        23Ø
C                                                                     24Ø
C THE FOLLOWING TWO STATEMENTS SET STORAGE LIMITS.                    25Ø
C     DIMENSION A(1ØØØØ),KA(6ØØØ)                                     26Ø
      MAXKV=6ØØØ                                                      27Ø
      IN=5                                                            28Ø
      OT=6                                                            29Ø
C                                                                     3ØØ
C TITLE AND CONTROL CARDS ARE READ  FOR CURRENT CASE.                 31Ø
C                                                                     32Ø
    1 READ (IN,4) HED                                                 33Ø
        IF (HED(1) .EQ. TITLE) GO TO 2                                34Ø
        IF (HED(1) .NE. END  ) GO TO 1                                35Ø
C END OF JOB                                                          36Ø
      STOP                                                            37Ø
C                                                                     38Ø
C ARRAY POINTERS ARE DETERMINED.                                      39Ø
C CORRESPONDENCE                                                      4ØØ
C   POINTER   N1   N2   N3   N4   N5   N6   N7   N8                   41Ø
C     ARRAY   VX   VY   VZ   KFA  AX   AY   AZ   KV                   42Ø
C                                                                     43Ø
    2 ERROR=.FALSE.                                                   44Ø
      READ (IN,7) MAXV,MAXS,DENS                                      45Ø
      WRITE (OT,6) HED,MAXV,MAXS,DENS                                 46Ø
      N1=1                                                            47Ø
      N2=N1+MAXV                                                      48Ø
      N3=N2+MAXV                                                      49Ø
      N4=1                                                            5ØØ
      N5=N3+MAXV                                                      51Ø
      N6=N5+MAXS                                                      52Ø
      N7=N6+MAXS                                                      53Ø
      N8=N4+MAXS                                                      54Ø
C                                                                     55Ø
C VERTEX COORDINATES AND FACE LISTS ARE INPUT                         56Ø
C                                                                     57Ø
C     CALL FACES ( KV,   KFA,  VX,   VY,   VZ  )                     58Ø
      CALL FACES (KA(N8),KA(N4),A(N1),A(N2),A(N3))                    59Ø
        IF (ERROR) GO TO 3                                            6ØØ
C                                                                     61Ø
C MASS PROPERTIES ARE CALCULATED                                      62Ø
C                                                                     63Ø
C     CALL PROPS (  KV,  KFA,  AX,   AY,   AZ,                       64Ø
C    .              VX,   VY,   VZ  )                                65Ø
      CALL PROPS (KA(N8),KA(N4),A(N5),A(N6),A(N7),                    66Ø
     .            A(N1),A(N2),A(N3))                                 67Ø
        IF (ERROR) GO TO 3                                            68Ø
C                                                                     69Ø
C AND PRINTED OUT.                                                    7ØØ
C                                                                     71Ø
      CALL MASSPR                                                     72Ø
      GO TO 1                                                         73Ø
C                                                                     74Ø
```

```
C ERROR ENCOUNTERED.  ERROR CODE PRINTED OUT.                          75Ø
C                                                                      76Ø
    3 WRITE (OT,5) HED,NERR                                            77Ø
      GO TO 1                                                          78Ø
C                                                                      79Ø
C FORMAT STATEMENTS                                                    8ØØ
C                                                                      81Ø
    4 FORMAT (8ØA1)                                                    82Ø
    5 FORMAT (1H1,8ØA1///27H ERROR ENCOUNTERED, CODE = ,I1)            83Ø
    6 FORMAT (1H1,8ØA1/// 7H MAXV =,I3/                                84Ø
     .                    7H MAXS =,I3/                               85Ø
     .                    7H DENS =,F1Ø.3)                            86Ø
    7 FORMAT (2I5,F1Ø.1Ø)                                              87Ø
      END                                                              88Ø
      SUBROUTINE FACES (KF,KFA,VX,VY,VZ)                               89Ø
C                                                                      9ØØ
C PURPOSE                                                              91Ø
C   TO READ VERTEX COORDINATES AND FACE LIST DATA AND SETUP            92Ø
C   DATA IN GLOBAL STORAGE.                                            93Ø
C                                                                      94Ø
C DESCRIPTION                                                          95Ø
C   A FACE BOUNDARY IS MADE UP OF A CLOSED SET OF NUMBERED             96Ø
C   POINTS (VERTICES) CONNECTED BY STRAIGHT LINES.                     97Ø
C   A FACE LIST, ENUMERATING THE VERTICES OF THE FACE TAKEN            98Ø
C   IN SUCCESSION AROUND THE BOUNDARY, IS THE DATA                     99Ø
C   REPRESENTATION OF THE FACE.                                       1ØØØ
C   A SURFACE OF THE SOLID MAY BE COMPOSED OF MORE THAN ONE           1Ø1Ø
C   FACE, AN INITIAL FACE AND SECONDARY FACE(S).  THEY ARE            1Ø2Ø
C   LINKED TOGETHER IN STORAGE BY A POINTER FROM ONE FACE             1Ø3Ø
C   TO ANOTHER.                                                       1Ø4Ø
C   THERE ARE NO PRACTICAL RESTRICTIONS ON THE NUMBER OF              1Ø5Ø
C   VERTICES PER FACE.  THE MAXIMUM NUMBER OF SURFACES IS             1Ø6Ø
C   CONTROLLED BY THE EXTENT OF AVAILABLE COMPUTER STORAGE            1Ø7Ø
C   AND IS ALSO DEPENDENT ON THE TOTAL NUMBER OF VERTICES.            1Ø8Ø
C                                                                     1Ø9Ø
      COMMON      MAXS,MAXV,                                          11ØØ
     .            AREA,WEIGHT,VOL,DENS,                              111Ø
     .            CG(3),IO(3),ICG(3),PR(3),PRCG(3),                  112Ø
     .            ERROR,NERR,                                        113Ø
     .            HED(8Ø),IN,OT,KT,                                  114Ø
     .            N8,MAXKV                                           115Ø
      INTEGER HED                                                    116Ø
      DIMENSION KF(1),KFA(1),VX(1),VY(1),VZ(1)                       117Ø
      LOGICAL ERROR                                                  118Ø
      INTEGER OT                                                     119Ø
C                                                                    12ØØ
C                                                                    121Ø
C VERTEX COORDINATES ARE READ IN AND PRINTED.                        122Ø
C                                                                    123Ø
      KT=5Ø                                                          124Ø
    1 READ (IN,14) IEND,NV,X,Y,Z                                     125Ø
        CALL COORDS (NV,X,Y,Z)                                       126Ø
        IF (NV .LE. Ø .OR. NV .GT. MAXV) GO TO 9                     127Ø
        VX(NV)=X                                                     128Ø
        VY(NV)=Y                                                     129Ø
        VZ(NV)=Z                                                     13ØØ
        IF (IEND .EQ. Ø) GO TO 1                                     131Ø
C                                                                    132Ø
C FACE POINTER ARRAY IS CLEARED.                                     133Ø
C                                                                    134Ø
      DO 2 I=1,MAXS                                                  135Ø
    2   KFA(I)=Ø                                                     136Ø
      KT=5Ø                                                          137Ø
      M1=1                                                           138Ø
    3 M2=M1+2                                                        139Ø
      M3=M1+16                                                       14ØØ
C                                                                    141Ø
C SURFACE ELEMENT DEFINITIONS ARE INPUT AND STORED.                  142Ø
C                                                                    143Ø
      READ (IN,15) IEND,KF(M1),(KF(M),M=M2,M3)                       144Ø
        NF=KF(M1)                                                    145Ø
        K=KFA(NF)                                                    146Ø
        KF(M1+1)=Ø                                                   147Ø
        IF (K .NE. Ø) GO TO 4                                        148Ø
C INITIAL (OR ONLY) FACE DEFINITION.                                 149Ø
        KFA(NF)=M1                                                   15ØØ
```

```
        GO TO 5                                                          151Ø
C SECONDARY FACE POINTER IS STORED.                                      152Ø
    4   KK=K                                                             153Ø
        K=KF(KK+1)                                                       154Ø
        IF (K .NE. Ø) GO TO 4                                            155Ø
        K=KF(KK+1)                                                       156Ø
        IF (K .NE. Ø) GO TO 4                                            157Ø
        KF(KK+1)=M1                                                      158Ø
C                                                                        159Ø
C CHECK IS MADE TO SEE IF DEFINITION IS COMPLETE.                        16ØØ
C                                                                        161Ø
    5   KV1=KF(M2)                                                       162Ø
        M2=M2+1                                                          163Ø
    6   DO  7 I=M2,M3                                                    164Ø
          KV=KF(I)                                                       165Ø
          IF (KV .EQ. KV1) GO TO 8                                       166Ø
          IF (KV .LE. Ø .OR. KV .GT. MAXV) GO TO 8                       167Ø
    7     CONTINUE                                                       168Ø
C                                                                        169Ø
C DEFINITION CONTINUED.                                                  17ØØ
C                                                                        171Ø
        M2=M3+1                                                          172Ø
        M3=M3+16                                                         173Ø
        READ (IN,15) IEND,(KF(M),M=M2,M3)                                174Ø
        GO TO 6                                                          175Ø
C                                                                        176Ø
C GETTING SET FOR NEXT SURFACE, PRINT CURRENT SET OF VERTICES.           177Ø
C                                                                        178Ø
    8 M2=M1+2                                                            179Ø
      M3=I                                                               18ØØ
      CALL FACVRT(NF,KF,M2,M3)                                           181Ø
          IF (KV .LE. Ø .OR. KV .GT. MAXV) GO TO 11                      182Ø
      M1=I+1                                                             183Ø
      IF (M1+N8 .GT. MAXKV) GO TO 12                                     184Ø
      IF (IEND .EQ. Ø) GO TO 3                                           185Ø
      RETURN                                                             186Ø
C                                                                        187Ø
C ERROR RETURN.                                                          188Ø
C                                                                        189Ø
C  VERTEX NUMBER OUT OF RANGE                                            19ØØ
    9 NERR=1                                                             191Ø
        GO TO 13                                                         192Ø
C  INPUT FACE NUMBER OUT OF RANGE                                        193Ø
   1Ø NERR=2                                                             194Ø
        GO TO 13                                                         195Ø
C  ILLEGAL VERTEX NUMBER.                                                196Ø
   11 NERR=3                                                             197Ø
        GO TO 13                                                         198Ø
C  STORAGE EXCEEDED.                                                     199Ø
   12 NERR=4                                                             2ØØØ
   13 ERROR=.TRUE.                                                       2Ø1Ø
      RETURN                                                             2Ø2Ø
C                                                                        2Ø3Ø
C FORMAT STATEMENTS                                                      2Ø4Ø
C                                                                        2Ø5Ø
   14 FORMAT (I1,I4,5X,3F1Ø.Ø)                                           2Ø6Ø
   15 FORMAT (I1,I4,15I5)                                                2Ø7Ø
      END                                                                2Ø8Ø
      SUBROUTINE COORDS (NV,X,Y,Z)                                       2Ø9Ø
C                                                                        21ØØ
C PURPOSE                                                                211Ø
C   PRINTS VERTEX COORDINATES, WITH PAGE CONTROL.                        212Ø
C                                                                        213Ø
C                                                                        214Ø
      COMMON      MAXS,MAXV,                                             215Ø
     .            AREA,WEIGHT,VOL,DENS,                                  216Ø
     .            CG(3),IO(3),ICG(3),PR(3),PRCG(3),                      217Ø
     .            ERROR,NERR,                                            218Ø
     .            HED(8Ø),IN,OT,KT                                       219Ø
      INTEGER OT                                                         22ØØ
      INTEGER HED                                                        221Ø
C                                                                        222Ø
C                                                                        223Ø
      IF (KT .LT. 5Ø) GO TO 1                                            224Ø
C TITLE AND HEADING.                                                     225Ø
      WRITE (OT,2) HED                                                   226Ø
```

```
      KT=Ø                                                              227Ø
      WRITE (OT,3)                                                      228Ø
C VERTEX COORDINATES                                                    229Ø
    1 KT=KT+1                                                           23ØØ
      WRITE (OT,4) NV,X,Y,Z                                             231Ø
      RETURN                                                            232Ø
C                                                                       233Ø
C FORMAT STATEMENTS.                                                    234Ø
C                                                                       235Ø
    2 FORMAT (1H1,8ØA1///)                                              236Ø
    3 FORMAT (7H VERTEX,9X,1HX,12X,1HY,12X,1HZ /)                       237Ø
    4 FORMAT (I6,3F13.2)                                                238Ø
      END                                                               239Ø
      SUBROUTINE FACVRT (NF,KV,M2,M3)                                   24ØØ
C                                                                       241Ø
C PURPOSE                                                               242Ø
C   PRINT FACE VERTICES, WITH PAGE CONTROL.                             243Ø
C                                                                       244Ø
C                                                                       245Ø
      COMMON      MAXS,MAXV,                                            246Ø
     .            AREA,WEIGHT,VOL,DENS,                                247Ø
     .            CG(3),IO(3),ICG(3),PR(3),PRCG(3),                    248Ø
     .            ERROR,NERR,                                          249Ø
     .            HED(8Ø),IN,OT,KT                                     25ØØ
      DIMENSION KV(1)                                                   251Ø
      INTEGER HED                                                       252Ø
      INTEGER OT                                                        253Ø
C                                                                       254Ø
C                                                                       255Ø
      IF (KT .LT. 5Ø ) GO TO 1                                          256Ø
C TITLE AND HEADING.                                                    257Ø
      WRITE (OT,2) HED                                                  258Ø
      WRITE (OT,3)                                                      259Ø
      KT=Ø                                                              26ØØ
C                                                                       261Ø
C FACE VERTICES.                                                        262Ø
C                                                                       263Ø
    1 KT=KT+1                                                           264Ø
      M4=MINØ(M3,M2+9)                                                  265Ø
      WRITE (OT,4) NF,(KV(M),M=M2,M4)                                   266Ø
      IF (M4 .EQ. M3) RETURN                                            267Ø
      M4=M2+1Ø                                                          268Ø
      WRITE (OT,5)     (KV(M),M=M4,M3)                                  269Ø
      RETURN                                                            27ØØ
C                                                                       271Ø
C FORMAT STATEMENTS.                                                    272Ø
C                                                                       273Ø
    2 FORMAT (1H1,8ØA1///)                                              274Ø
    3 FORMAT (25H FACE     VERTICES  . . . /)                           275Ø
    4 FORMAT (/I5,5H  -  ,1ØI5)                                         276Ø
    5 FORMAT (1ØX,1ØI5)                                                 277Ø
      END                                                               278Ø
      SUBROUTINE MASSPR                                                 279Ø
C                                                                       28ØØ
C PURPOSE                                                               281Ø
C   PRINT MASS PROPERTIES OF A SOLID.                                   282Ø
C                                                                       283Ø
C DESCRIPTION                                                           284Ø
C   PRINTS                                                              285Ø
C   DENSITY,AREA,VOLUME,WEIGHT                                          286Ø
C   CENTER OF GRAVITY COORDINATES                                       287Ø
C   COORDINATES OF MASS MOMENTS OF INERTIA W/R TO ORIGIN                288Ø
C   COORDINATES OF MASS MOMENTS OF INERTIA W/R TO CG                    289Ø
C   COORDINATES OF PROD MOMENTS OF INERTIA W/R TO ORIGIN                29ØØ
C   COORDINATES OF PROD MOMENTS OF INERTIA W/R TO CG                    291Ø
C                                                                       292Ø
      COMMON      MAXS,MAXV,                                            293Ø
     .            AREA,WEIGHT,VOL,DENS,                                294Ø
     .            CG(3),IO(3),ICG(3),PR(3),PRCG(3),                    295Ø
     .            ERROR,NERR,                                          296Ø
     .            HED(8Ø),IN,OT,KT                                     297Ø
      INTEGER HED                                                       298Ø
      INTEGER OT                                                        299Ø
      REAL IO,ICG                                                       3ØØØ
C                                                                       3Ø1Ø
C                                                                       3Ø2Ø
```

```
      WRITE (OT,1) HED                                                  3Ø3Ø
      WRITE (OT,2) DENS,AREA,VOL,WEIGHT                                 3Ø4Ø
      WRITE (OT,3)                                                      3Ø5Ø
      WRITE (OT,4) CG                                                   3Ø6Ø
      WRITE (OT,5) IO                                                   3Ø7Ø
      WRITE (OT,6) ICG                                                  3Ø8Ø
      WRITE (OT,8)                                                      3Ø9Ø
      WRITE (OT,7) PR                                                   31ØØ
      WRITE (OT,6) PRCG                                                 311Ø
      RETURN                                                            312Ø
C                                                                       313Ø
C FORMAT STATEMENTS.                                                    314Ø
C                                                                       315Ø
    1 FORMAT (1H1,8ØA1///)                                              316Ø
    2 FORMAT (1ØX,7HDENSITY,9X,4HAREA,1ØX,6HVOLUME,9X,                  317Ø
     .           6HWEIGHT//2X,4F15.5///)                                318Ø
    3 FORMAT (13X,4HAXES,8X,6HX-COMP,9X,6HY-COMP,9X,                    319Ø
     .                      6HZ-COMP /)                                32ØØ
    4 FORMAT (1X,16H  CG       COORD,3F15.5/)                           321Ø
    5 FORMAT (1X,16HMOMENT     COORD,3F15.5 )                           322Ø
    6 FORMAT (1X,16H              CG,3F15.5/)                           323Ø
    7 FORMAT (1X,16HPROD MOM   COORD,3F15.5)                            324Ø
    8 FORMAT (//13X,4HAXES,8X,7HXY-COMP,8X,7HYZ-COMP,8X,                325Ø
     .7HZX-COMP/)                                                       326Ø
      END                                                               327Ø
C SOLID POLYHEDRON MEASURES                                             328Ø
C -------------------------                                            329Ø
C                                                                       33ØØ
C                                                                       331Ø
C PURPOSE                                                               332Ø
C   TO CALCULATE MASS PROPERTIES OF THREE DIMENSIONAL SOLIDS            333Ø
C   BY A SURFACE INTEGRATION TECHNIQUE USING FOUR-POINT                 334Ø
C   GUASSIAN QUADRATURE OVER TRIANGLES.                                 335Ø
C                                                                       336Ø
C SOLID                                                                 337Ø
C   DEFINED BY A SET OF SURFACES, WHERE EACH SURFACE IS                 338Ø
C   COMPOSED OF ONE OR MORE COPLANAR CLOSED POLYGONAL                   339Ø
C   PATCHES CALLED FACES.  DISCONNECTED FACES AND FACES WITH            34ØØ
C   ONE OR MORE HOLES IN THEM MAY BE USED, IF NECESSARY,                341Ø
C   TO REPRESENT THE SOLID.                                             342Ø
C                                                                       343Ø
C FACE                                                                  344Ø
C   A FACE IS REPRESENTED BY THE SET OF VERTEX POINTS OF THE            345Ø
C   POLYGONAL SURFACE PATCH, ORDERED BY STARTING WITH AN                346Ø
C   ARBITRARY VERTEX, AND PROCEEDING POINT BY POINT AROUND              347Ø
C   THE BOUNDARY UNTIL THE FIRST POINT IS REACHED.                      348Ø
C                                                                       349Ø
C FACE LIST                                                             35ØØ
C   THE LIST OF THE SET OF VERTEX NUMBERS, ORDERED AS ABOVE             351Ø
C   FOR A FACE, IS CALLED A FACE LIST, WHICH IS THE DATA                352Ø
C   REPRESENTATION OF A FACE.  A FACE LIST HAS THE FORM                 353Ø
C        NSF,NA,NB,NC,...,NI,...,NA                                     354Ø
C   WHERE NSF IS THE SURFACE NUMBER AND NA,NB,NC,NI ARE                 355Ø
C   VERTEX NUMBERS.  NOTE THAT THE FACE LIST IS A CLOSED                356Ø
C   LIST, STARTING AND ENDING WITH THE FIRST POINT SELECTED.            357Ø
C   FOR ANY SURFACE WITH ONE OR MORE HOLES IN                           358Ø
C   IT, ANY INTERIOR BOUNDARY (HOLE) IS REPRESENTED BY A                359Ø
C   FACE LIST WITH VERTICES ENUMERATED IN A SEQUENCE                    36ØØ
C   OPPOSITE TO THAT GIVEN FOR AN EXTERIOR BOUNDARY.                    361Ø
C                                                                       362Ø
C INITIAL, SECONDARY FACE LISTS                                         363Ø
C   WHEN MULTIPLE FACE LISTS ARE NEEDED TO DEFINE A SURFACE,            364Ø
C   (IE, A SURFACE WITH A HOLE OR ONE WITH DISCONNECTED BUT             365Ø
C   COPLANAR PARTS), THE FIRST LIST TO APPEAR IN THE DATA IS            366Ø
C   CALLED THE INITIAL FACE LIST, AND ANY OTHER IS CALLED A             367Ø
C   SECONDARY FACE LIST.  THESE LISTS WILL HAVE A COMMON                368Ø
C   SURFACE NUMBER AND WILL BE LINKED TOGETHER IN STORAGE BY            369Ø
C   THE DATA ENTRY ROUTINE.                                             37ØØ
C                                                                       371Ø
C GLOBAL DATA ARRAYS                                                    372Ø
C   FACE LIST DATA IS STORED COMPACTLY IN A LARGE OPEN-ENDED            373Ø
C   DATA ARRAY KV.                                                      374Ø
C   A LINKAGE ARRAY, KFA, SETUP BY THE DATA ENTRY ROUTINE,              375Ø
C   CONTAINS POINTERS TO THE INITIAL FACE LIST (IN KV) FOR              376Ø
C   EACH SURFACE, ORDERED BY SURFACE NUMBER.  IF L1, L2, L3             377Ø
C   ARE LOCATIONS IN KV OF FACE LISTS FOR SURFACES 1, 2, 3,             378Ø
```

```
C   THEN THE PROGRAM WOULD SET                                          3790
C        KFA(1)=L1                                                      3800
C        KFA(2)=L2                                                      3810
C        KFA(3)=L3                                                      3820
C   A POINTER VALUE OF ZERO (KFA(I)=0) INDICATES SURFACE I              3830
C   WAS NOT UTILIZED IN DEFINING THE SOLID.  HENCE GAPS MAY             3840
C   OCCUR IN THE SURFACE NUMBERS.  ALSO, THE ORDER OF DATA              3850
C   ENTRY OF THE FACE LISTS CAN BE ENTIRELY ARBITRARY,                  3860
C   BECAUSE OF THE AFORE-MENTIONED LINKAGE MECHANISM.                   3870
C                                                                       3880
C LINKAGE TO SECONDARY FACE LISTS                                       3890
C   A FACE LIST IN THE ARRAY KV HAS THE FORM                            3900
C        NSF,LSI,NA,NB,NC,...,NI,NJ,NK,...,NA                           3910
C   WHERE NSF IC THE SURFACE NUMBER, NA,NB,... ARE VERTEX               3920
C   NUMBERS, AND LSI IS A LINK TO A SECONDARY FACE LIST.                3930
C   LSI=0 INDICATES THIS IS THE LAST FACE LIST FOR A SURFACE.           3940
C   NOTE, IT MAY BE THE ONLY LIST.                                      3950
C                                                                       3960
C INTEGRATION TECHNIQUE                                                 3970
C   THE ALGORITHM IS PERFORMED BY TWO SUBROUTINES, PROPS AND            3980
C   SRFINT.                                                             3990
C   SURFACES ARE PROCESSED ONE AT A TIME AND THE MASS                   4000
C   PROPERTY CALCULATIONS ARE ACCUMULATED DURING THE                    4010
C   PROCESSING.                                                         4020
C   SURFACE PROCESSING INVOLVES STARTING WITH THE INITIAL               4030
C   FACE, AND PROCEDING THROUGH THE SECONDARY FACES, IF ANY.            4040
C   SURFACE AREA VECTOR COMPONENTS ARE SUMMED FOR EACH FACE             4050
C   OF THE SURFACE.  THE AREA VECTOR, WHOSE MAGNITUDE IS THE            4060
C   AREA OF THE SURFACE, IS THEN COMPUTED FROM THE                      4070
C   COMPONENTS AND IS ACCUMULATED IN THE TOTAL SURFACE                  4080
C   AREA OF THE SOLID.                                                  4090
C   IN THE FACE PROCESSING, FACES ARE DIVIDED INTO ALL                  4100
C   UNIQUE TRIANGLES HAVING AS VERTICES THE THE FIRST ONE OF            4110
C   THE LIST AND ANY OTHER TWO ADJACENT VERTICES OF THE FACE            4120
C   POLYGON.  FOUR-POINT GAUSSIAN QUADRATURE IS APPLIED TO              4130
C   EACH TRIANGLE, AND THE RESULTS ARE ACCUMULATED IN THE               4140
C   TOTAL MASS PROPERTIES.  THUS EACH TRIANGLE OF EACH FACE             4150
C   OF EACH SURFACE HAS A CONTRIBUTION IN THE FINAL RESULTS.            4160
C                                                                       4170
C GLOBAL NOMENCLATURE                                                   4180
C   AREA   = SURFACE AREA OF SOLID                                      4190
C   AX,AY,                                                              4200
C      AZ  = SURFACE AREA VECTOR  COMPONENTS ARRAYS                     4210
C   CG     = CENTER OF GRAVITY COORDINATE ARRAY                         4220
C   DENS   = MASS DENSITY OF SOLID                                      4230
C   ERROR  = LOGICAL ERROR INDICATOR                                    4240
C   ICG    = COORDS OF MASS MOMENTS, WITH RESPECT TO CG                 4250
C   IO     = COORDS OF MASS MOMENTS, WITH RESPECT TO ORIGIN             4260
C   KFA    = POINTER ARRAY.  KFA(I) POINTS TO LOCATION IN KV            4270
C            OF INITIAL FACE LIST FOR SURFACE I.                        4280
C   KV     = FACE LISTS STORAGE ARRAY                                   4290
C   MAXS   = NUMBER OF SURFACES ON SOLID                                4300
C   MAXV   = NUMBER OF VERTEX POINTS USED IN DEFINING SOLID             4310
C   NERR   = ERROR CODE                                                 4320
C   NS     = CURRENT SURFACE NUMBER                                     4330
C   PR     = COORDS OF PROD MOMENTS, WITH RESPECT TO ORIGIN             4340
C   PRCG   = COORDS OF PROD MOMENTS, WITH RESPECT TO CG                 4350
C   VOL    = VOLUME OF SOLID                                            4360
C   VX,VY,                                                              4370
C      VZ  = VERTEX COORDINATE ARRAYS                                   4380
C   WEIGHT = WEIGHT OF SOLID                                            4390
C                                                                       4400
      SUBROUTINE PROPS (KV,KFA,AX,AY,AZ,VX,VY,VZ)                       4410
C                                                                       4420
C PURPOSE                                                               4430
C   CALCULATES MASS PROPERTIES OF A GENERAL THREE                       4440
C   DIMENSIONAL SOLID USING FOUR POINT GAUSSIAN QUADRATURE              4450
C   OVER TRIANGLES.                                                     4460
C                                                                       4470
C DESCRIPTION                                                           4480
C   CALCULATES CONTRIBUTION OF EACH SURFACE TO MASS                     4490
C   PROPERTIES INTEGRALS.  INTEGRATION IS PERFORMED IN                  4500
C   SUBROUTINE SRFINT.  MASS PROPERTIES CALCULATIONS ARE                4510
C   THEN COMPLETED IN THIS ROUTINE.  DATA IS ASSUMED TO HAVE            4520
C   BEEN PREVIOUSLY SETUP IN STORAGE BY A DATA ENTRY ROUTINE.           4530
C                                                                       4540
```

```
C LOCAL NOMENCLATURE                                                    455Ø
C   NS      = CURRENT SURFACE BEING PROCESSED                           456Ø
C   LA      = POINTER TO FACE LIST (INITIAL OR SECONDARY) DATA          457Ø
C                                                                       458Ø
      COMMON       MAXS,MAXV,                                           459Ø
     .             AREA,WEIGHT,VOL,DENS,                                46ØØ
     .             CG(3),IO(3),ICG(3),PR(3),PRCG(3),                    461Ø
     .             ERROR,NERR                                           462Ø
      DIMENSION KV(1),KFA(1),VX(1),VY(1),VZ(1)                          463Ø
      DIMENSION AX(1),AY(1),AZ(1)                                       464Ø
      LOGICAL ERROR                                                     465Ø
      REAL IO,ICG                                                       466Ø
C                                                                       467Ø
C CLEARING OF ACCUMULATION PARAMETERS.                                  468Ø
C                                                                       469Ø
      AREA=Ø.                                                           47ØØ
      VOL=Ø.                                                            471Ø
      DO 1 J=1,MAXS                                                     472Ø
        AX(J)=Ø.                                                        473Ø
        AY(J)=Ø.                                                        474Ø
    1   AZ(J)=Ø.                                                        475Ø
      DO 2 J=1,3                                                        476Ø
        CG(J)=Ø.                                                        477Ø
        IO(J)=Ø.                                                        478Ø
    2   PR(J)=Ø.                                                        479Ø
C                                                                       48ØØ
C EACH SURFACE IS PROCESSED.                                            481Ø
C                                                                       482Ø
      DO 4 NS=1,MAXS                                                    483Ø
C LA IS A POINTER TO THE FACE LIST DATA                                 484Ø
        LA=KFA(NS)                                                      485Ø
        IF (LA .EQ. Ø) GO TO 4                                          486Ø
C CONTRIBUTION OF EACH FACE OF CURRENT SURFACE IS CALCULATED            487Ø
    3   IF (KV(LA) .NE. NS) GO TO 5                                     488Ø
        CALL SRFINT (KV(LA+2),AX,AY,AZ,VX,VY,VZ,NS)                     489Ø
C CHECK IS MADE FOR SECONDARY FACE LIST.                                49ØØ
        LA=KV(LA+1)                                                     491Ø
        IF (LA .NE. Ø) GO TO 3                                          492Ø
C AREA SUMMATION.                                                       493Ø
        AREA=AREA+SQRT(AX(NS)**2+AY(NS)**2+AZ(NS)**2)                   494Ø
    4   CONTINUE                                                        495Ø
C                                                                       496Ø
C MASS PROPERTY CALCULATIONS ARE COMPLETED.                             497Ø
C                                                                       498Ø
      VOL=VOL/3.Ø                                                       499Ø
      WEIGHT=VOL*DENS                                                   5ØØØ
      VOL2=VOL*2.                                                       5Ø1Ø
      VOL3=VOL/3.Ø                                                      5Ø2Ø
      DENS2=DENS/2.Ø                                                    5Ø3Ø
      DENS3=DENS/3.Ø                                                    5Ø4Ø
      CG(1)=CG(1)/VOL2                                                  5Ø5Ø
      CG(2)=CG(2)/VOL2                                                  5Ø6Ø
      CG(3)=CG(3)/VOL2                                                  5Ø7Ø
      IO(1)=IO(1)*DENS3                                                 5Ø8Ø
      IO(2)=IO(2)*DENS3                                                 5Ø9Ø
      IO(3)=IO(3)*DENS3                                                 51ØØ
      ICG(1)=IO(1)-(CG(2)**2+CG(3)**2)*WEIGHT                           511Ø
      ICG(2)=IO(2)-(CG(1)**2+CG(3)**2)*WEIGHT                           512Ø
      ICG(3)=IO(3)-(CG(1)**2+CG(2)**2)*WEIGHT                           513Ø
      PR(1)=PR(1)*DENS2                                                 514Ø
      PR(2)=PR(2)*DENS2                                                 515Ø
      PR(3)=PR(3)*DENS2                                                 516Ø
      PRCG(1)=PR(1)-CG(1)*CG(2)*WEIGHT                                  517Ø
      PRCG(2)=PR(2)-CG(2)*CG(3)*WEIGHT                                  518Ø
      PRCG(3)=PR(3)-CG(3)*CG(1)*WEIGHT                                  519Ø
      RETURN                                                            52ØØ
C                                                                       521Ø
C ERROR RETURN, DATA REDUNDANCY CHECK FAILED.                           522Ø
C                                                                       523Ø
    5 ERROR=.TRUE.                                                      524Ø
      NERR=5                                                            525Ø
      RETURN                                                            526Ø
      END                                                               527Ø
      SUBROUTINE SRFINT (KV,AX,AY,AZ,VX,VY,VZ,NS)                       528Ø
C                                                                       529Ø
C PURPOSE                                                               53ØØ
```

```
C   TO CALCULATE SURFACE INTEGRALS OVER A FACE OF A SOLID.                5310
C                                                                          5320
C DESCRIPTION                                                              5330
C   DIVIDES FACE INTO TRIANGLES HAVING AS VERTICES THE                     5340
C   FIRST VERTEX OF THE POLYGON AND TWO ADJACENT VERTICES                  5350
C   OF THE POLYGON.  NUMERICAL INTEGRATION IS PERFORMED                    5360
C   USING FOUR-POINT GAUSSIAN QUADRATURE OVER THE TRIANGLES.               5370
C   RESULTS ARE ACCUMULATED TO GET TOTALS.                                 5380
C                                                                          5390
C LOCAL NOMENCLATURE                                                       5400
C   ANX,                                                                   5410
C    ANY,                                                                  5420
C     ANZ  = AREA VECTOR COMPONENTS FOR SINGLE TRIANGLE                    5430
C   CX,CY,                                                                 5440
C       CZ = 1ST,2ND OR 3RD POWERS OF QUADRATURE COORDS,                   5450
C            WEIGHTED AND AREA NORMALIZED                                  5460
C   PX,PY,                                                                 5470
C       PZ = QUADRATURE POINTS COORDINATE ARRAYS                           5480
C   W      = WEIGHTING FACTORS ARRAY                                       5490
C   X,Y,Z  = FACE TRIANGLE COORDINATE ARRAYS                               5500
C                                                                          5510
      COMMON      MAXS,MAXV,                                               5520
     .            AREA,WEIGHT,VOL,DENS,                                    5530
     .            CG(3),IO(3),ICG(3),PR(3),PRCG(3)                         5540
      DIMENSION KV(1),AX(1),AY(1),AZ(1),VX(1),VY(1),VZ(1)                  5550
      DIMENSION X(3),Y(3),Z(3),PX(4),PY(4),PZ(4),W(4)                      5560
      REAL IO,ICG                                                          5570
C                                                                          5580
C W(I) ARE WEIGHING FACTORS FOR 4-POINT GAUSSIAN QUADRATURE                5590
C OVER TRIANGLES.                                                          5600
C                                                                          5610
      W(1)=-.5625                                                          5620
      W(2)=.520833333333333                                                5630
      W(3)=.520833333333333                                                5640
      W(4)=.520833333333333                                                5650
C                                                                          5660
C THE POLYGON IS SEGMENTED INTO TRIANGLES, EACH HAVING THE                 5670
C FIRST VERTEX OF THE POLYGON AS THE FIRST POINT OF THE                    5680
C TRIANGLE.                                                                5690
C                                                                          5700
      K=KV(1)                                                              5710
      X(1)=VX(K)                                                           5720
      Y(1)=VY(K)                                                           5730
      Z(1)=VZ(K)                                                           5740
      DO 3 L=2,1000                                                        5750
        K=KV(L+1)                                                          5760
C TESTING FOR END OF DEFINITION.                                           5770
        IF (K .EQ. KV(1)) GO TO 4                                          5780
        X(3)=VX(K)                                                         5790
        Y(3)=VY(K)                                                         5800
        Z(3)=VZ(K)                                                         5810
        K=KV(L)                                                            5820
        X(2)=VX(K)                                                         5830
        Y(2)=VY(K)                                                         5840
        Z(2)=VZ(K)                                                         5850
C                                                                          5860
C DATA IS NOW IN TERMS OF A SINGLE TRIANGLE.                               5870
C                                                                          5880
        PX(1)=(X(1)+X(2)+X(3))/3.                                          5890
        PY(1)=(Y(1)+Y(2)+Y(3))/3.                                          5900
        PZ(1)=(Z(1)+Z(2)+Z(3))/3.                                          5910
        DX=.6*PX(1)                                                        5920
        DY=.6*PY(1)                                                        5930
        DZ=.6*PZ(1)                                                        5940
        DO 1 I=1,3                                                         5950
          K=I+1                                                            5960
          PX(K)=DX+.4*X(I)                                                 5970
          PY(K)=DY+.4*Y(I)                                                 5980
          PZ(K)=DZ+.4*Z(I)                                                 5990
    1     CONTINUE                                                         6000
C .                                                                        6010
C AREA VECTORS CALCULATED.                                                 6020
C                                                                          6030
        ANX=.5*(Z(1)*(Y(2)-Y(3))+Z(2)*(Y(3)-Y(1))                          6040
     .         +Z(3)*(Y(1)-Y(2)))                                          6050
        ANY=.5*(X(1)*(Z(2)-Z(3))+X(2)*(Z(3)-Z(1))                          6060
```

```
     .         +X(3)*(Z(1)-Z(2)))                                    6070
        ANZ=.5*(Y(1)*(X(2)-X(3))+Y(2)*(X(3)-X(1))                    6080
     .         +Y(3)*(X(1)-X(2)))                                    6090
        AX(NS)=AX(NS)+ANX                                            6100
        AY(NS)=AY(NS)+ANY                                            6110
        AZ(NS)=AZ(NS)+ANZ                                            6120
C                                                                    6130
C CALCULATION OF MASS PROPERTIES                                     6140
C                                                                    6150
        DO  2 I=1,4                                                  6160
C 1ST POWER OF INTEGRATION COORDINATES                               6170
          CX=ANX*PX(I)*W(I)                                          6180
          CY=ANY*PY(I)*W(I)                                          6190
          CZ=ANZ*PZ(I)*W(I)                                          6200
          VOL=VOL+(CX+CY+CZ)                                         6210
C 2ND POWER OF INTEGRATION COORDINATES                               6220
          CX=CX*PX(I)                                                6230
          CY=CY*PY(I)                                                6240
          CZ=CZ*PZ(I)                                                6250
          CG(1)=CG(1)+CX                                             6260
          CG(2)=CG(2)+CY                                             6270
          CG(3)=CG(3)+CZ                                             6280
          PR(1)=PR(1)+CX*PY(I)                                       6290
          PR(2)=PR(2)+CY*PZ(I)                                       6300
          PR(3)=PR(3)+CZ*PX(I)                                       6310
C 3RD POWER OF INTEGRATION COORDINATES                               6320
          CX=CX*PX(I)                                                6330
          CY=CY*PY(I)                                                6340
          CZ=CZ*PZ(I)                                                6350
          IO(1)=IO(1)+CY+CZ                                          6360
          IO(2)=IO(2)+CZ+CX                                          6370
          IO(3)=IO(3)+CX+CY                                          6380
    2     CONTINUE                                                   6390
    3   CONTINUE                                                     6400
C                                                                    6410
    4 CONTINUE                                                       6420
      RETURN                                                         6430
      END                                                            6440
```

ALGORITHM 551
A Fortran Subroutine for the L_1 Solution of Overdetermined Systems of Linear Equations [F4]

NABIH N. ABDELMALEK
National Research Council, Ottawa, Canada

Key Words and Phrases: overdetermined system of linear equations, discrete linear L_1 approximation, linear programming, dual simplex algorithm, triangular decomposition
CR Categories: 5.13, 5.41
Language: Fortran

DESCRIPTION

This algorithm is a Fortran implementation for the procedure developed in [1] and [2]. It solves an overdetermined system of linear equations of the form

$$Ca = f,$$

in the L_1 norm. C is a given real n-by-m matrix of rank $k \leq m \leq n$ and f is a given real n-vector.

The L_1 solution to this system is the m-vector a^* which minimizes the L_1 norm

$$z = \sum_{i=1}^{n} |r_i|,$$

where r_i is the ith residual and is given by

$$r_i = \sum_{j=1}^{m} c_{ij}a_j - f_i.$$

This subroutine uses a dual simplex method and a suitable triangular decomposition method to the basis matrix. In [2] the testing of this subroutine, as well as numerical results and comments, are given.

REFERENCES

1. ABDELMALEK, N.N. An efficient method for the discrete linear L_1 approximation problem. *Math. Comput. 29* (1975), 844–850.
2. ABDELMALEK, N.N. L_1 solution of overdetermined systems of linear equations. *ACM Trans. Math. Softw. 6*, 2 (June 1980), 220–227.

ALGORITHM

```
C     THIS PROGRAM IS A DRIVER FOR THE SUBROUTINE  L1
C WHICH SOLVES AN OVERDETERMINED SYSTEM OF LINEAR EQUATIONS
C IN THE L1 NORM, USING A DUAL SIMPLEX METHOD.
C     THE OVERDETERMINED SYSTEM HAS THE FORM  CA=F
C   C IS A GIVEN REAL N BY M MATRIX OF RANK K, K.LE.M.LE.N
C   F IS A GIVEN REAL N-VECTOR.
```

Received 12 March 1974; revised 17 March 1977 and 19 June 1979; accepted 10 July 1979.

Author's address: Division of Electrical Engineering, National Research Council of Canada, Ottawa, Ont., Canada, KIA OR8.

```
C  A IS THE SOLUTION M-VECTOR.
C
C
C
      DIMENSION AA(252,27)
      DIMENSION FAY(8,25Ø),EF(25Ø)
      DIMENSION CT(25,25Ø),F(25Ø),R(25Ø),A(25),BINV(25,25)
      DIMENSION P(35Ø),IC(25Ø),IB(25Ø),Y(25Ø),TH(25Ø),IR(25)
      DIMENSION U(25),V(25),W(25),GINV(25,25),VB(25)
      DIMENSION BV(25),IBOUND(25Ø),ICBAS(25),IRBAS(25),ZC(25Ø)
      DIMENSION F1(5Ø),F2(5Ø),BA(6,15)
      INTEGER S(25Ø)
C
  1Ø1 FORMAT(1ØH1EXAMPLE  ,I3,8H SIZE   ,I3,8H  BY    ,I3)
  1Ø2 FORMAT(4ØHØ   R.H.S.   OF THE SYSTEM              )
  1Ø3 FORMAT(4ØHØTRANSPOSE OF COEFFICIENT MATRIX        )
  1Ø4 FORMAT(17HØEXECUTION TIME =,F12.5,13HSECONDS      )
  1Ø5 FORMAT(9HØL1 NORM=,F1Ø.5,6H,RANK=,I4,6H,ITER=,I4,5H,IND=,I4)
  1Ø6 FORMAT(4ØHØTHE ANSWER :  A(I)   OR   X(I)         )
  1Ø7 FORMAT(4ØHØRESIDUALS  R(J)    OR E(J)             )
  1Ø8 FORMAT(4ØHØIC(I)                                  )
  1Ø9 FORMAT(4ØHØIR(I)                                  )
  11Ø FORMAT(1H ,1ØF12.5)
  111 FORMAT(1H ,2ØI4)
  115 FORMAT(9HØL1 NORM=,F1Ø.5,6H,RANK=,F3.Ø,6H,ITER=,F3.Ø,5H,IND=,F2.Ø)
  12Ø FORMAT(1H ,8F15.5)
  125 FORMAT(5F1Ø.6)
  126 FORMAT(6F1Ø.6)
C
      CALL SUNDER
      PREC=1.E-6
      EPS=1.E-4
      TOLER=1.ØE-4
      MM=25
      MM2=MM+2
      MMM=(MM*(MM+3))/2
      NN=25Ø
      NN2=NN+2
      IEXMPL=Ø
    1 IEXMPL=IEXMPL+1
      GO TO (2,4,5,6,7,8,9,13,17,19,2Ø,21,22,26,27,28,29,32,35,1ØØ),
     *IEXMPL
C EXAMPLE 1.
    2 N=2Ø1
      DX=Ø.Ø2
      DO 3 I=1,N
      X1=DX*FLOAT(I-1)
      X2=X1*X1
      X3=X2*X1
      X4=X2*X2
      X5=X3*X2
      X6=X3*X3
      X7=X3*X4
      FAY(1,I)=1.
      FAY(2,I)=X1
      FAY(3,I)=X2
      FAY(4,I)=X3
      FAY(5,I)=X4
      FAY(6,I)=X5
      FAY(7,I)=X6
    3 EF(I)= (EXP(-X1))*SIN(X1)
      M=1
      GO TO 1Ø
    4 M=2
      GO TO 1Ø
    5 M=3
      GO TO 1Ø
    6 M=4
      GO TO 1Ø
    7 M=5
      GO TO 1Ø
    8 M=6
      GO TO 1Ø
    9 M=7
   1Ø DO 12 J=1,N
      F(J)=EF(J)
```

```
      DO 11 I=1,M
   11 CT(I,J)=FAY(I,J)
   12 CONTINUE
      GO TO 9Ø
C EXAMPLE 2.
   13 N=1ØØ
      N2=5Ø
      M=6
      READ(1,125)(F1(I), I=1,N2)
      READ(1,125)(F2(I), I=1,N2)
      DO 14 J=1,N2
      F(J)=F1(J)
      JN2=J+N2
   14 F(JN2)=F2(J)
      DO 16 J=1,1Ø
      D=Ø.1*FLOAT(J)
      DD=D*D
        DO 15 I=1,1Ø
        E=Ø.1*FLOAT(I)
        EE=E*E
        K=1Ø*(J-1)+I
        CT(1,K)=1.Ø
        CT(2,K)=E
        CT(3,K)=D
        CT(4,K)=E*D
        CT(5,K)=EE
        CT(6,K)=DD
   15   CONTINUE
   16 CONTINUE
      GO TO 9Ø
C EXAMPLE 3.
   17 N=51
      M=8
      DO 18 I=1,N
      D=Ø.Ø2*FLOAT(I-1)
      DD=D*D
      DDD=D*DD
      E1=D-Ø.1
      E2=D-Ø.2
      E3=D-Ø.4
      E4=D-Ø.7
      E13=E1*E1*E1
      E23=E2*E2*E2
      E33=E3*E3*E3
      E43=E4*E4*E4
      CT(1,I)=1.Ø
      CT(2,I)=D
      CT(3,I)=DD
      CT(4,I)=DDD
      IF(E1.GT.Ø.Ø) CT(5,I)=E13
      IF(E1.LE.Ø.Ø) CT(5,I)=Ø.
      IF(E2.GT.Ø.Ø) CT(6,I)=E23
      IF(E2.LE.Ø.Ø) CT(6,I)=Ø.
      IF(E3.GT.Ø.Ø) CT(7,I)=E33
      IF(E3.LE.Ø.Ø) CT(7,I)=Ø.
      IF(E4.GT.Ø.Ø) CT(8,I)=E43
      IF(E4.LE.Ø.Ø) CT(8,I)=Ø.
   18 F(I)= SQRT(D)
      GO TO 9Ø
C EXAMPLE 4.
   19 M=6
      PI=4.Ø*ATAN(1.Ø)
      XØ1=PI/2.Ø
      XØ2=XØ1*XØ1
      XØ3=XØ2*XØ1
      XØ4=XØ2*XØ2
      N=23
      GO TO 23
   2Ø N=53
      GO TO 23
   21 N=1Ø3
      GO TO 23
   22 N=2Ø3
   23 N2=N-2
      N3=N-3
      DX=PI/FLOAT(N3)
```

```
      DO 24 I=1,N2
      X1=-XØ1+DX*FLOAT(I-1)
      X2=X1*X1
      X3=X2*X1
      X4=X2*X2
      X5=X3*X2
      CT(1,I)=1.Ø
      CT(2,I)=X1
      CT(3,I)=X2
      CT(4,I)=X3
      CT(5,I)=X4
      CT(6,I)=X5
   24 F(I)= SIN(X1)
      G=-1.Ø
      DO 25 I=1,2
      IF(I.EQ.2) G=1.Ø
      J=N2+I
      CT(1,J)=Ø.Ø
      CT(2,J)=1ØØØ.Ø
      CT(3,J)=2ØØØ.Ø*G*XØ1
      CT(4,J)=3ØØØ.Ø*XØ2
      CT(5,J)=4ØØØ.Ø*G*XØ3
      CT(6,J)=5ØØØ.Ø*XØ4
   25 F(J)=Ø.Ø
      GO TO 9Ø
C EXAMPLE 5.
   26 M=11
      D=2Ø.Ø
      DX=1.Ø/D
      N=21
      GO TO 3Ø
   27 D=5Ø.Ø
      DX=1.Ø/D
      N=51
      GO TO 3Ø
   28 D=1ØØ.Ø
      DX=1.Ø/D
      N=1Ø1
      GO TO 3Ø
   29 D=2ØØ.Ø
      DX=1.Ø/D
      N=2Ø1
   3Ø DO 31 I=1,N
      X1=DX*FLOAT(I-1)
      X2=X1+X1
      X3=X1+X2
      X4=X2+X2
      X5=X2+X3
      CT(1,I)=1.Ø
      CT(2,I)= SIN(X1)
      CT(3,I)= COS(X1)
      CT(4,I)= SIN(X2)
      CT(5,I)= COS(X2)
      CT(6,I)= SIN(X3)
      CT(7,I)= COS(X3)
      CT(8,I)= SIN(X4)
      CT(9,I)= COS(X4)
      CT(1Ø,I)= SIN(X5)
      CT(11,I)= COS(X5)
      G= EXP(X1)
      G1= EXP(Ø.5)
      F(I)=G1
      IF(G.LT.G1) F(I)=G
   31 CONTINUE
      GO TO 9Ø
C EXAMPLE 6.
   32 M=5
      M1=M+1
      N=15
      DO 34 J=1,N
      READ(1,126)(BA(I,J),I=1,M1)
        DO 33 I=1,M
   33   CT(I,J)=BA(I,J)
      F(J)=BA(M1,J)
   34 CONTINUE
      GO TO 9Ø
```

```
C EXAMPLE 7
   35 M=5
      N=51
      DX=Ø.Ø2
      DO 36 I=1,N
      X1=DX*FLOAT(I-1)
      X2=X1*X1
      X3=X2*X1
      X4=X2*X2
      CT(1,I)=1.Ø
      CT(2,I)=X1
      CT(3,I)=X2
      CT(4,I)=X3
      CT(5,I)=X4
      GX=Ø.Ø
      IF(X1.GT.Ø.93) GX=5.Ø
      F(I)=X1*(1.Ø+X1*(1.Ø+X1*(1.Ø+X1)))+GX
   36 CONTINUE
C
   9Ø WRITE (3,1Ø1)IEXMPL,N,M
C     WRITE(3,1Ø2)
C     WRITE(3,11Ø)(F(J),J=1,N)
C     WRITE(3,1Ø3)
C     DO 93 I=1,M
C  93 WRITE(3,11Ø)(CT(I,J),J=1,N)
      CALL SETCLK
      CALL L1(M,N,MM,MMM,NN,CT,F,PREC,EPS,IC,IR,IB,
     *U,V,W,Y,TH,P,GINV,VB,IRANK,ITER,R,A,Z,IND)
      CALL RDCLK(TX)
      WRITE(3,1Ø8)
      WRITE(3,111)(IC(I),I=1,M)
      WRITE(3,1Ø9)
      WRITE(3,111)(IR(I),I=1,M)
      WRITE(3,1Ø7)
      WRITE(3,11Ø)(R(J),J=1,N)
      WRITE(3,1Ø6)
      WRITE(3,12Ø)(A(I),I=1,M)
      WRITE(3,1Ø5)Z,IRANK,ITER,IND
      WRITE(3,1Ø4)TX
      GO TO 1
  1ØØ STOP
      END
```

```
      SUBROUTINE PSLV(ID,K,MM,MMM,P,B,X)                              PSLVØØ1Ø
C     THIS SUBROUTINE SOLVES THE SQUARE NON-SINGULAR SYSTEM           PSLVØØ2Ø
C OF LINEAR EQUATIONS                                                 PSLVØØ3Ø
C                              P*X=B,                                 PSLVØØ4Ø
C OR THE SQUARE NON-SINGULAR SYSTEM OF LINEAR EQUATIONS               PSLVØØ5Ø
C                          P(TRANSPOSE)*X=B,                          PSLVØØ6Ø
C WHERE P IS AN UPPER TRIANGULAR MATRIX, B IS THE RIGHT HAND          PSLVØØ7Ø
C SIDE VECTOR AND X IS THE SOLUTION VECTOR.                           PSLVØØ8Ø
C     THE INPUT DATA TO THE SUBROUTINE.                               PSLVØØ9Ø
C ID     AN INTEGER INDICATOR SPECIFIED BY THE USER.                  PSLVØ1ØØ
C        IF ID=1 THE EQUATION P*X=B IS SOLVED.                        PSLVØ11Ø
C        IF ID= ANY INTEGER OTHER THAN 1, THE EQUATION                PSLVØ12Ø
C        P(TRANSPOSE)*X=B IS SOLVED.                                  PSLVØ13Ø
C K      AN INTEGER = THE NUMBER OF EQUATIONS OF THE GIVEN            PSLVØ14Ø
C        SYSTEM.                                                      PSLVØ15Ø
C MM     AN INTEGER GREATER THAN OR EQUAL TO K.                       PSLVØ16Ø
C MMM    AN INTEGER = (MM*(MM+3))/2                                   PSLVØ17Ø
C P      AN MMM-VECTOR. ITS FIRST (K+1) ELEMENTS CONTAIN THE          PSLVØ18Ø
C        FIRST K ELEMENTS OF ROW 1 OF THE UPPER TRIANGULAR            PSLVØ19Ø
C        MATRIX PLUS AN EXTRA LOCATION TO THE RIGHT. ITS              PSLVØ2ØØ
C        NEXT K ELEMENTS CONTAIN THE (K-1) ELEMENTS OF ROW 2          PSLVØ21Ø
C        OF THE UPPER TRIANGULAR MATRIX PLUS AN EXTRA                 PSLVØ22Ø
C        LOCATION TO THE RIGHT, ..., ETC.                             PSLVØ23Ø
C B      AN MM-VECTOR. ITS FIRST K ELEMENTS CONTAIN THE               PSLVØ24Ø
C        R.H.S. VECTOR OF THE GIVEN SYSTEM.                           PSLVØ25Ø
C     THE OUTPUT OF THE SUBROUTINE.                                   PSLVØ26Ø
C X      AN MM-VECTOR. ON EXIT, ITS FIRST K ELEMENTS CONTAIN          PSLVØ27Ø
C        THE SOLUTION TO THE GIVEN SYSTEM.                            PSLVØ28Ø
      DOUBLE PRECISION S,SA,SB                                        PSLVØ29Ø
      DIMENSION P(MMM),B(MM),X(MM)                                    PSLVØ3ØØ
      IF(ID.NE.1) GO TO 3                                             PSLVØ31Ø
C SOLUTION OF THE UPPER TRIANGULAR SYSTEM.                            PSLVØ32Ø
```

```
      L=(K-1)+(K*(K+1))/2                                          PSLVØ33Ø
      X(K)=B(K)/P(L)                                               PSLVØ34Ø
      IF(K.EQ.1) RETURN                                            PSLVØ35Ø
      KD=3                                                          PSLV036
      KM1=K-1                                                      PSLVØ37Ø
      DO 2 I=1,KM1                                                 PSLVØ38Ø
        J=K-I                                                      PSLVØ39Ø
        L=L-KD                                                     PSLVØ4ØØ
        KD=KD+1                                                    PSLVØ41Ø
        S=B(J)                                                     PSLVØ42Ø
        LL=L                                                       PSLVØ43Ø
        JJ=J                                                       PSLVØ44Ø
        DO 1 KK=1,I                                                PSLVØ45Ø
          LL=LL+1                                                  PSLVØ46Ø
          SA=P(LL)                                                 PSLVØ47Ø
          JJ=JJ+1                                                  PSLVØ48Ø
          SB=X(JJ)                                                 PSLVØ49Ø
    1     S=S-SA*SB                                                PSLVØ5ØØ
        X(J)=S/P(L)                                                PSLVØ51Ø
    2   CONTINUE                                                   PSLVØ52Ø
      RETURN                                                       PSLVØ53Ø
C SOLUTION OF THE LOWER TRIANGULAR SYSTEM.                         PSLVØ54Ø
    3 X(1)=B(1)/P(1)                                               PSLVØ55Ø
      IF(K.EQ.1) RETURN                                            PSLVØ56Ø
      L=1                                                          PSLVØ57Ø
      KD=K+1                                                       PSLVØ58Ø
      DO 5 I=2,K                                                   PSLVØ59Ø
        L=L+KD                                                     PSLVØ6ØØ
        KD=KD-1                                                    PSLVØ61Ø
        S=B(I)                                                     PSLVØ62Ø
        KK=I                                                       PSLVØ63Ø
        KKM1=I-1                                                   PSLVØ64Ø
        KKD=K                                                      PSLVØ65Ø
        DO 4 J=1,KKM1                                              PSLVØ66Ø
          SA=P(KK)                                                 PSLVØ67Ø
          SB=X(J)                                                  PSLVØ68Ø
          S=S-SA*SB                                                PSLVØ69Ø
          KK=KK+KKD                                                PSLVØ7ØØ
          KKD=KKD-1                                                PSLVØ71Ø
    4     CONTINUE                                                 PSLVØ72Ø
    5   X(I)=S/P(L)                                                PSLVØ73Ø
      RETURN                                                       PSLVØ74Ø
      END                                                          PSLVØ75Ø
```

```
      SUBROUTINE L1(M,N,MM,MMM,NN,CT,F,PREC,EPS,IC,IR,IB,          L1  ØØ1Ø
     *UF,BP,XP,T,ALFA,P,GINV,VB,IRANK,ITER,R,A,Z,IND)              L1  ØØ2Ø
C     THIS SUBROUTINE SOLVES AN OVERDETERMINED SYSTEM OF           L1  ØØ3Ø
C LINEAR EQUATIONS IN THE L1 NORM BY USING A DUAL SIMPLEX          L1  ØØ4Ø
C ALGORITHM TO THE LINEAR PROGRAMMING FORMULATION OF THE           L1  ØØ5Ø
C GIVEN PROBLEM. IN THIS ALGORITHM CERTAIN INTERMEDIATE            L1  ØØ6Ø
C SIMPLEX ITERATIONS ARE SKIPPED.                                  L1  ØØ7Ø
C     FOR PURPOSE OF NUMERICAL STABILITY, THIS SUBROUTINE          L1  ØØ8Ø
C USES A TRIANGULAR DECOMPOSITION TO THE BASIS MATRIX.             L1  ØØ9Ø
C     THE SYSTEM OF LINEAR EQUATIONS HAS THE FORM                  L1  Ø1ØØ
C                          C*A=F,                                  L1  Ø11Ø
C WHERE C IS A GIVEN REAL N BY M MATRIX OF RANK K.LE.M.LE.N         L1  Ø12Ø
C AND F IS A GIVEN REAL N-VECTOR.                                  L1  Ø13Ø
C     THE PROBLEM TO BE SOLVED IS TO CALCULATE THE ELEMENTS        L1  Ø14Ø
C OF THE M-VECTOR  A*  WHICH GIVES THE MINIMUM NORM Z.             L1  Ø15Ø
C               Z= ABS(R(1)) + ABS(R(2)) + ... + ABS(R(N)) ,       L1  Ø16Ø
C WHERE R(I) IS THE ITH RESIDUAL AND IS GIVEN BY                   L1  Ø17Ø
C   R(I)=C(I,1)*A(1)+C(I,2)*A(2)+ ... +C(I,M)*A(M)-F(I).          L1  Ø18Ø
C     SUBROUTINE L1 IS COMPLETELY SELF CONTAINED (CONSISTS         L1  Ø19Ø
C OF TWO SUBROUTINES L1   AND PSLV).                               L1  Ø2ØØ
C     THE INPUT DATA TO THE SUBROUTINE.                            L1  Ø21Ø
C M       THE NUMBER OF COLUMNS OF MATRIX C.                       L1  Ø22Ø
C N       THE NUMBER OF ROWS OF MATRIX C.                          L1  Ø23Ø
C MM      AN INTEGER GREATER THAN OR EQUAL TO M.                   L1  Ø24Ø
C MMM     AN INTEGER = (MM*(MM+3))/2 .                             L1  Ø25Ø
C NN      AN INTEGER GREATER THAN OR EQUAL TO N.                   L1  Ø26Ø
C CT      A MATRIX OF DIMENSIONS MM BY NN. ON ENTRY, ITS FIRST     L1  Ø27Ø
C         M ROWS AND FIRST N COLUMNS CONTAIN THE TRANSPOSE OF      L1  Ø28Ø
C         MATRIX C IN THE GIVEN SYSTEM C*A=F.                      L1  Ø29Ø
C F       AN NN-VECTOR. ON ENTRY, ITS FIRST N ELEMENTS CONTAIN     L1  Ø3ØØ
C         THE R.H.S. OF THE GIVEN SYSTEM C*A=F.                    L1  Ø31Ø
```

```
C PREC   THE ROUND-OFF LEVEL OF THE COMPUTER. FOR THE IBM          L1  Ø32Ø
C        36Ø/67 COMPUTER, PREC IS ABOUT 1.E-6 AND 1.E-16 FOR       L1  Ø33Ø
C        SINGLE AND DOUBLE PRECISION RESPECTIVELY.                 L1  Ø34Ø
C EPS    A SPECIFIED TOLERANCE SUCH THAT A CALCULATED NUMBER       L1  Ø35Ø
C        X IS CONSIDERED ZERO IF  ABS(X) <EPS. FOR THE IBM 36Ø/67  L1  Ø36Ø
C        COMPUTER, EPS IS USUALLY TAKEN 1.E-4 AND 1.E-11           L1  Ø37Ø
C        RESPECTIVELY.                                             L1  Ø38Ø
C     THE RESULTS OF THE PROBLEM.                                  L1  Ø39Ø
C IRANK  THE CALCULATED RANK OF MATRIX C.                         L1  Ø4ØØ
C ITER   THE NUMBER OF ITERATIONS WHICH THE SOLUTION NEEDED.      L1  Ø41Ø
C A      AN MM-VECTOR. ITS FIRST M ELEMENTS ARE THE SOLUTION      L1  Ø42Ø
C        VECTOR A*.                                                L1  Ø43Ø
C R      AN NN-VECTOR. ITS FIRST N ELEMENTS CONTAIN THE           L1  Ø44Ø
C        THE RESIDUAL VECTOR R=C*A-F.                              L1  Ø45Ø
C Z      THE MINIMUM L1 NORM OF THE RESIDUAL VECTOR R.            L1  Ø46Ø
C IND    A RETURN INDICATOR. IND=Ø INDICATES THAT THE             L1  Ø47Ø
C        SOLUTION VECTOR A* IS UNIQUE. IND=1, INDICATES THAT       L1  Ø48Ø
C        A* IS MOST PROBABLY NOT UNIQUE. IND=-1 INDICATES          L1  Ø49Ø
C        PREMATURE TERMINATION OF THE CALCULATION DUE TO           L1  Ø5ØØ
C        VERY ILL-CONDITIONING OF MATRIX C.                        L1  Ø51Ø
C     THE MEANING OF THE OTHER PARAMETERS.                         L1  Ø52Ø
C GINV   AN MM-SQUARE MATRIX. ITS FIRST IRANK COLUMNS AND         L1  Ø53Ø
C        FIRST IRANK ROWS CONTAIN THE INVERSE OF THE INITIAL       L1  Ø54Ø
C        BASIS MATRIX AND ITS UPDATE.                              L1  Ø55Ø
C VB     AN MM-VECTOR. ITS FIRST IRANK ELEMENTS CONTAIN THE        L1  Ø56Ø
C        INITIAL BASIC SOLUTION AND ITS UPDATE.                    L1  Ø57Ø
C T      AN NN-VECTOR. ITS FIRST N ELEMENTS CONTAIN THE            L1  Ø58Ø
C        ELEMENTS OF THE ROW IN THE SIMPLEX TABLEAU, THAT          L1  Ø59Ø
C        CORRESPONDS TO THE COLUMN WHICH LEAVES THE BASIS.         L1  Ø6ØØ
C ALFA   AN NN-VECTOR. ITS FIRST N ELEMENTS CONTAIN THE           L1  Ø61Ø
C        RATIOS:  ALFA(J) = R(J)/T(J).                             L1  Ø62Ø
C IC     AN NN-VECTOR. ITS FIRST N ELEMENTS CONTAIN THE           L1  Ø63Ø
C        COLUMN INDICES OF MATRIX CT.                              L1  Ø64Ø
C IR     AN MM VECTOR. ITS FIRST IRANK ELEMENTS CONTAIN THE       L1  Ø65Ø
C        ROW INDICES OF THE LINEARLY INDEPENDENT ROWS OF CT.       L1  Ø66Ø
C IB     A SIGN NN-VECTOR. ITS FIRST N ELEMENTS HAVE THE          L1  Ø67Ø
C        VALUES +1 OR -1. IB(J)=+1 INDICATES THAT COLUMN J         L1  Ø68Ø
C        IN THE TABLEAU IS AT ITS LOWER BOUND Ø. IB(J)=-1          L1  Ø69Ø
C        INDICATES THAT COLUMN J IS AT ITS UPPER BOUND 2.          L1  Ø7ØØ
C P      AN MMM-VECTOR. ITS FIRST ((IRANK*(IRANK+3))/2)-1         L1  Ø71Ø
C        ELEMENTS CONTAIN THE (IRANK*(IRANK+1))/2 ELEMENTS OF      L1  Ø72Ø
C        THE UPPER TRIANGULAR MATRIX + EXTRA (IRANK-1) WORKING     L1  Ø73Ø
C        LOCATIONS. SEE THE COMMENTS IN SUBROUTINE PSLV.           L1  Ø74Ø
C BP     AN MM-VECTOR. ITS FIRST IRANK ELEMENTS ARE THE           L1  Ø75Ø
C        R.H.S. OF THE TRIANGULAR EQUATIONS AS P*XP=BP.            L1  Ø76Ø
C XP     AN MM-VECTOR . ITS FIRST IRANK ELEMENTS ARE THE          L1  Ø77Ø
C        SOLUTION OF THE TRIANGULAR EQUATIONS AS P*XP=BP.          L1  Ø78Ø
C UF     AN MM-WORKING VECTOR.                                    L1  Ø79Ø
      DOUBLE PRECISION S,SA,SB                                     L1  Ø8ØØ
      DIMENSION CT(MM,NN),F(NN),A(MM),GINV(MM,MM),P(MMM)           L1  Ø81Ø
      DIMENSION IC(NN),IB(NN),R(NN),T(NN),ALFA(NN),IR(MM)          L1  Ø82Ø
      DIMENSION UF(MM),BP(MM),XP(MM),VB(MM)                        L1  Ø83Ø
      IND=Ø                                                        L1  Ø84Ø
      TPEPS=2.+EPS                                                 L1  Ø85Ø
      NMM=(M*(M+3))/2                                              L1  Ø86Ø
      IRANK=M                                                      L1  Ø87Ø
      ITER=Ø                                                       L1  Ø88Ø
      DO 1 J=1,N                                                   L1  Ø89Ø
        IB(J)=1                                                    L1  Ø9ØØ
    1   IC(J)=J                                                    L1  Ø91Ø
      DO 3 J=1,M                                                   L1  Ø92Ø
        IR(J)=J                                                    L1  Ø93Ø
        A(J)=Ø.                                                    L1  Ø94Ø
        DO 2 I=1,M                                                 L1  Ø95Ø
    2     GINV(I,J)=Ø.                                             L1  Ø96Ø
    3   GINV(J,J)=1.                                               L1  Ø97Ø
      IOUT=Ø                                                       L1  Ø98Ø
C PART 1 OF THE ALGORITHM.                                         L1  Ø99Ø
    4 IOUT=IOUT+1                                                  L1  1ØØØ
      IF(IOUT.GT.IRANK) GO TO 16                                   L1  1Ø1Ø
      PIV=Ø.                                                       L1  1Ø2Ø
      DO 6 J=IOUT,N                                                L1  1Ø3Ø
        ICJ=IC(J)                                                  L1  1Ø4Ø
        DO 5 I=IOUT,IRANK                                          L1  1Ø5Ø
          D=CT(I,ICJ)                                              L1  1Ø6Ø
          IF(D.LT.Ø.Ø) D=-D                                        L1  1Ø7Ø
```

```
            IF(D.LE.PIV) GO TO 5                                    L1  1Ø8Ø
            LI=I                                                    L1  1Ø9Ø
            JIN=ICJ                                                 L1  11ØØ
            LJ=J                                                    L1  111Ø
            PIV=D                                                   L1  112Ø
        5     CONTINUE                                              L1  113Ø
        6   CONTINUE                                                L1  114Ø
C DETECTION OF RANK DEFICIENCY.                                     L1  115Ø
          IF(PIV.GT.EPS) GO TO 7                                    L1  116Ø
          IRANK=IOUT-1                                              L1  117Ø
          IND=1                                                     L1  118Ø
          GO TO 16                                                  L1  119Ø
        7 IF(LI.EQ.IOUT) GO TO 1Ø                                   L1  12ØØ
          DO 8 J=1,N                                                L1  121Ø
            G=CT(LI,J)                                              L1  122Ø
            CT(LI,J)=CT(IOUT,J)                                     L1  123Ø
        8   CT(IOUT,J)=G                                           L1  124Ø
          K=IR(LI)                                                  L1  125Ø
          IR(LI)=IR(IOUT)                                           L1  126Ø
          IR(IOUT)=K                                                L1  127Ø
          IF(IOUT.EQ.1) GO TO 1Ø                                    L1  128Ø
          K=IOUT-1                                                  L1  129Ø
          DO 9 J=1,K                                                L1  13ØØ
            D=GINV(LI,J)                                            L1  131Ø
            GINV(LI,J)=GINV(IOUT,J)                                 L1  132Ø
        9   GINV(IOUT,J)=D                                          L1  133Ø
C A GAUSS-JORDAN ELIMINATION STEP.                                  L1  134Ø
       1Ø PIVOT=CT(IOUT,JIN)                                        L1  135Ø
          DO 11 J=1,N                                               L1  136Ø
       11   CT(IOUT,J)=CT(IOUT,J)/PIVOT                            L1  137Ø
          DO 12 J=1,IOUT                                            L1  138Ø
       12   GINV(IOUT,J)=GINV(IOUT,J)/PIVOT                        L1  139Ø
          DO 15 I=1,IRANK                                           L1  14ØØ
            IF(I.EQ.IOUT) GO TO 15                                  L1  141Ø
            D=CT(I,JIN)                                             L1  142Ø
            DO 13 J=1,N                                             L1  143Ø
       13     CT(I,J)=CT(I,J)-D*CT(IOUT,J)                         L1  144Ø
            DO 14 J=1,IOUT                                          L1  145Ø
       14     GINV(I,J)=GINV(I,J)-D*GINV(IOUT,J)                   L1  146Ø
       15   CONTINUE                                                L1  147Ø
          ITER=ITER+1                                               L1  148Ø
          K=IC(LJ)                                                  L1  149Ø
          IC(LJ)=IC(IOUT)                                           L1  15ØØ
          IC(IOUT)=K                                                L1  151Ø
          GO TO 4                                                   L1  152Ø
C PART 2 OF THE ALGORITHM.                                          L1  153Ø
       16 IRANK1=IRANK+1                                            L1  154Ø
          IRNKM1=IRANK-1                                            L1  155Ø
C INITIAL RESIDUALS AND INITIAL BASIC SOLUTION.                     L1  156Ø
          DO 17 J=1,IRANK                                           L1  157Ø
            ICJ=IC(J)                                               L1  158Ø
            R(ICJ)=Ø.                                               L1  159Ø
       17   UF(J)=F(ICJ)                                            L1  16ØØ
          DO 19 J=IRANK1,N                                          L1  161Ø
            ICJ=IC(J)                                               L1  162Ø
            S=-F(ICJ)                                               L1  163Ø
            DO 18 I=1,IRANK                                         L1  164Ø
              SA=UF(I)                                              L1  165Ø
              SB=CT(I,ICJ)                                          L1  166Ø
       18     S=S+SA*SB                                            L1  167Ø
            R(ICJ)=S                                                L1  168Ø
            IF(S.GE.Ø.Ø) GO TO 19                                   L1  169Ø
            IB(ICJ)=-1                                              L1  17ØØ
       19   CONTINUE                                                L1  171Ø
          DO 21 I=1,IRANK                                           L1  172Ø
            S=Ø.                                                    L1  173Ø
            DO 2Ø J=1,N                                             L1  174Ø
              SA=CT(I,J)                                            L1  175Ø
              IF(IB(J).EQ.(-1)) SA=-SA                              L1  176Ø
       2Ø     S=S+SA                                               L1  177Ø
       21   VB(I)=S                                                 L1  178Ø
C INITIALIZING THE TRIANGULAR MATRIX.                               L1  179Ø
          DO 22 I=1,NMM                                             L1  18ØØ
       22   P(I)=Ø.                                                 L1  181Ø
          K=1                                                       L1  182Ø
          KD=IRANK1                                                 L1  183Ø
```

```
      DO 23 I=1,IRANK                                          L1  184Ø
        P(K)=1.                                                L1  185Ø
        K=K+KD                                                 L1  186Ø
   23   KD=KD-1                                                L1  187Ø
C DETERMINE THE VECTOR WHICH LEAVES THE BASIS.                 L1  188Ø
   24 IVO=Ø                                                    L1  189Ø
      CALL PSLV(1,IRANK,MM,MMM,P,VB,XP)                        L1  19ØØ
      G=1.                                                     L1  191Ø
      DO 27 I=1,IRANK                                          L1  192Ø
        E=XP(I)                                                L1  193Ø
        IF(E.LT.(-EPS)) GO TO 25                               L1  194Ø
        IF(E.LE.TPEPS) GO TO 27                                L1  195Ø
        D=2.Ø-E                                                L1  196Ø
        IF(D.GE.G) GO TO 27                                    L1  197Ø
        IVO=1                                                  L1  198Ø
        GO TO 26                                               L1  199Ø
   25   D=E                                                    L1  2ØØØ
        IF(D.GE.G) GO TO 27                                    L1  2Ø1Ø
        IVO=-1                                                 L1  2Ø2Ø
   26   G=D                                                    L1  2Ø3Ø
        IOUT=I                                                 L1  2Ø4Ø
        XB=E                                                   L1  2Ø5Ø
   27   CONTINUE                                               L1  2Ø6Ø
      IF(IVO.EQ.Ø) GO TO 66                                    L1  2Ø7Ø
C CALCULATION OF ROW (IOUT) IN THE TABLEAU.                    L1  2Ø8Ø
      ICIOUT=IC(IOUT)                                          L1  2Ø9Ø
      T(ICIOUT)=1.                                             L1  21ØØ
      BXB=XB                                                   L1  211Ø
      DO 28 I=1,IRANK                                          L1  212Ø
   28   BP(I)=Ø.                                               L1  213Ø
      BP(IOUT)=1.                                              L1  214Ø
      CALL PSLV(2,IRANK,MM,MMM,P,BP,XP)                        L1  215Ø
      ALFMX=Ø.Ø                                                L1  216Ø
      DO 31 J=IRANK1,N                                         L1  217Ø
        ICJ=IC(J)                                              L1  218Ø
        ALFA(ICJ)=Ø.                                           L1  219Ø
        S=Ø.                                                   L1  22ØØ
        DO 29 I=IOUT,IRANK                                     L1  221Ø
          SA=XP(I)                                             L1  222Ø
          SB=CT(I,ICJ)                                         L1  223Ø
   29     S=S+SA*SB                                            L1  224Ø
        E=S                                                    L1  225Ø
        T(ICJ)=E                                               L1  226Ø
        IF(E.LT.EPS.AND.E.GT.(-EPS)) GO TO 31                  L1  227Ø
        D=R(ICJ)                                               L1  228Ø
        IF(D.NE.Ø.Ø) GO TO 3Ø                                  L1  229Ø
        D=PREC*PREC*FLOAT(IB(ICJ))                             L1  23ØØ
   3Ø   ALFA(ICJ)=D/E                                          L1  231Ø
        GG=ALFA(ICJ)                                           L1  232Ø
        IF(GG.LT.Ø.Ø) GG=-GG                                   L1  233Ø
        IF(GG.LE.ALFMX) GO TO 31                               L1  234Ø
        ALFMX=GG                                               L1  235Ø
   31   CONTINUE                                               L1  236Ø
      PIVOTO=1.                                                L1  237Ø
      ITEST=Ø                                                  L1  238Ø
C DETERMINE THE VECTOR WHICH ENTERS THE BASIS.                 L1  239Ø
      GG=ALFMX+ALFMX                                           L1  24ØØ
   32 ALFMX=GG                                                 L1  241Ø
      ALFMN=-GG                                                L1  242Ø
      DO 35 J=IRANK1,N                                         L1  243Ø
        ICJ=IC(J)                                              L1  244Ø
        E=ALFA(ICJ)                                            L1  245Ø
        D=E*FLOAT(IVO)                                         L1  246Ø
        IF(D.LE.Ø.Ø) GO TO 35                                  L1  247Ø
        IF(IVO.EQ.1) GO TO 33                                  L1  248Ø
        IF(E.LE.ALFMN) GO TO 35                                L1  249Ø
        ALFMN=E                                                L1  25ØØ
        GO TO 34                                               L1  251Ø
   33   IF(E.GE.ALFMX) GO TO 35                                L1  252Ø
        ALFMX=E                                                L1  253Ø
   34   JIN=J                                                  L1  254Ø
        ITEST=1                                                L1  255Ø
   35   CONTINUE                                               L1  256Ø
      IF(ITEST.EQ.1) GO TO 36                                  L1  257Ø
C NO FEASIBLE SOLUTION HAS BEEN FOUND.                         L1  258Ø
```

```
      IND=-1                                                          L1  259Ø
      GO TO 66                                                        L1  26ØØ
   36 ICJIN=IC(JIN)                                                   L1  261Ø
      PIVOT=T(ICJIN)                                                  L1  262Ø
      ALPHA=ALFA(ICJIN)                                               L1  263Ø
      PIVOTN=PIVOT/PIVOTO                                             L1  264Ø
      IF(XB.GT.TPEPS) GO TO 37                                        L1  265Ø
      IF(PIVOTN.GT.Ø.) GO TO 39                                       L1  266Ø
      GO TO 41                                                        L1  267Ø
   37 DO 38 I=1,IRANK                                                 L1  268Ø
        E=CT(I,ICIOUT)                                                L1  269Ø
   38   VB(I)=VB(I)-E-E                                               L1  27ØØ
      E=T(ICIOUT)                                                     L1  271Ø
      BXB=BXB-E-E                                                     L1  272Ø
      IB(ICIOUT)=-1                                                   L1  273Ø
      IF(PIVOTN.GT.Ø.) GO TO 41                                       L1  274Ø
   39 DO 4Ø I=1,IRANK                                                 L1  275Ø
        E=CT(I,ICJIN)                                                 L1  276Ø
   4Ø   VB(I)=VB(I)+E+E                                               L1  277Ø
      E=T(ICJIN)                                                      L1  278Ø
      BXB=BXB+E+E                                                     L1  279Ø
      IB(ICJIN)=1                                                     L1  28ØØ
   41 XB=BXB/PIVOT                                                    L1  281Ø
      IF(XB.GE.(-EPS).AND.XB.LE.TPEPS) GO TO 42                       L1  282Ø
      ITEST=Ø                                                         L1  283Ø
   42 ALFA(ICJIN)=Ø.                                                  L1  284Ø
      IF(ITEST.EQ.1) GO TO 43                                         L1  285Ø
      PIVOTO=PIVOT                                                    L1  286Ø
      ICIOUT=ICJIN                                                    L1  287Ø
      GO TO 32                                                        L1  288Ø
   43 R(ICJIN)=Ø.                                                     L1  289Ø
      ITER=ITER+1                                                     L1  29ØØ
      IF(IOUT.EQ.IRANK) GO TO 46                                      L1  291Ø
C UPDATING MATRIX (P,GINV,VB,CT).                                     L1  292Ø
      DO 45 J=IOUT,IRNKM1                                             L1  293Ø
        K=J                                                           L1  294Ø
        K1=K+1                                                        L1  295Ø
        KD=IRANK                                                      L1  296Ø
        L=IC(K)                                                       L1  297Ø
        IC(K)=IC(K1)                                                  L1  298Ø
        IC(K1)=L                                                      L1  299Ø
        DO 44 I=1,K1                                                  L1  3ØØØ
          P(K)=P(K+1)                                                 L1  3Ø1Ø
          K=K+KD                                                      L1  3Ø2Ø
   44     KD=KD-1                                                     L1  3Ø3Ø
   45   CONTINUE                                                      L1  3Ø4Ø
   46 L=IC(IRANK)                                                     L1  3Ø5Ø
      IC(IRANK)=IC(JIN)                                               L1  3Ø6Ø
      IC(JIN)=L                                                       L1  3Ø7Ø
      K=IRANK                                                         L1  3Ø8Ø
      KD=IRANK                                                        L1  3Ø9Ø
      DO 47 I=1,IRANK                                                 L1  31ØØ
        P(K)=CT(I,ICJIN)                                              L1  311Ø
        K=K+KD                                                        L1  312Ø
   47   KD=KD-1                                                       L1  313Ø
      IF(IOUT.EQ.IRANK) GO TO 58                                      L1  314Ø
      DO 57 I=IOUT,IRNKM1                                             L1  315Ø
        II=I                                                          L1  316Ø
        I1=I+1                                                        L1  317Ø
        K=Ø                                                           L1  318Ø
        KD=IRANK1                                                     L1  319Ø
        DO 48 J=1,II                                                  L1  32ØØ
          K=K+KD                                                      L1  321Ø
   48     KD=KD-1                                                     L1  322Ø
        KK=K                                                          L1  323Ø
        KL=K-KD                                                       L1  324Ø
        L=KL                                                          L1  325Ø
        G=P(K)                                                        L1  326Ø
        D=P(L)                                                        L1  327Ø
        IF(G.LT.Ø.Ø) G=-G                                             L1  328Ø
        IF(D.LT.Ø.Ø) D=-D                                             L1  329Ø
        IF(G.LE.D) GO TO 53                                           L1  33ØØ
        DO 49 J=II,IRANK                                              L1  331Ø
          E=P(K)                                                      L1  332Ø
          P(K)=P(L)                                                   L1  333Ø
          P(L)=E                                                      L1  334Ø
```

```
            K=K+1                                                    L1  335Ø
   49       L=L+1                                                    L1  336Ø
            J=IR(I)                                                  L1  337Ø
            IR(I)=IR(I1)                                             L1  338Ø
            IR(I1)=J                                                 L1  339Ø
            DO 5Ø J=1,N                                              L1  34ØØ
              E=CT(I,J)                                              L1  341Ø
              CT(I,J)=CT(I1,J)                                       L1  342Ø
   5Ø       CT(I1,J)=E                                               L1  343Ø
            DO 51 J=1,IRANK                                          L1  344Ø
              E=GINV(I,J)                                            L1  345Ø
              GINV(I,J)=GINV(I1,J)                                   L1  346Ø
   51       GINV(I1,J)=E                                             L1  347Ø
            DO 52 J=1,IRANK                                          L1  348Ø
              E=GINV(J,I)                                            L1  349Ø
              GINV(J,I)=GINV(J,I1)                                   L1  35ØØ
   52       GINV(J,I1)=E                                             L1  351Ø
            E=VB(I)                                                  L1  352Ø
            VB(I)=VB(I1)                                             L1  353Ø
            VB(I1)=E                                                 L1  354Ø
   53       E=P(KK)/P(KL)                                            L1  355Ø
            IF(E.LT.PREC.AND.E.GT.(-PREC)) GO TO 57                  L1  356Ø
            K=KK                                                     L1  357Ø
            L=KL                                                     L1  358Ø
            DO 54 J=II,IRANK                                         L1  359Ø
              P(K)=P(K)-E*P(L)                                       L1  36ØØ
              K=K+1                                                  L1  361Ø
   54       L=L+1                                                    L1  362Ø
            DO 55 J=1,N                                              L1  363Ø
   55       CT(I1,J)=CT(I1,J)-E*CT(I,J)                              L1  364Ø
            DO 56 J=1,IRANK                                          L1  365Ø
   56       GINV(I1,J)=GINV(I1,J)-E*GINV(I,J)                        L1  366Ø
            VB(I1)=VB(I1)-E*VB(I)                                    L1  367Ø
   57       CONTINUE                                                 L1  368Ø
   58    DO 59 J=1,IRANK                                             L1  369Ø
            ICJ=IC(J)                                                L1  37ØØ
   59       UF(J)=F(ICJ)                                             L1  371Ø
         IF(ALPHA.GT.(1.Ø).OR.ALPHA.LT.(-1.Ø)) GO TO 61              L1  372Ø
         DO 6Ø J=IRANK1,N                                            L1  373Ø
            ICJ=IC(J)                                                L1  374Ø
   6Ø       R(ICJ)=R(ICJ)-ALPHA*T(ICJ)                               L1  375Ø
         GO TO 64                                                    L1  376Ø
   61    CALL PSLV(2,IRANK,MM,MMM,P,UF,XP)                           L1  377Ø
         DO 63 J=IRANK1,N                                            L1  378Ø
            ICJ=IC(J)                                                L1  379Ø
            S=-F(ICJ)                                                L1  38ØØ
            DO 62 I=1,IRANK                                          L1  381Ø
              SA=XP(I)                                               L1  382Ø
              SB=CT(I,ICJ)                                           L1  383Ø
   62         S=S+SA*SB                                              L1  384Ø
   63       R(ICJ)=S                                                 L1  385Ø
   64    DO 65 J=IRANK1,N                                            L1  386Ø
            ICJ=IC(J)                                                L1  387Ø
            D=R(ICJ)*FLOAT(IB(ICJ))                                  L1  388Ø
            IF(D.GE.Ø.Ø) GO TO 65                                    L1  389Ø
            R(ICJ)=Ø.Ø                                               L1  39ØØ
   65       CONTINUE                                                 L1  391Ø
         GO TO 24                                                    L1  392Ø
C CALCULATING THE ANSWER OF THE PROBLEM.                             L1  393Ø
   66    CALL PSLV(2,IRANK,MM,MMM,P,UF,VB)                           L1  394Ø
         DO 68 I=1,IRANK                                             L1  395Ø
            S=Ø.                                                     L1  396Ø
            DO 67 K=1,IRANK                                          L1  397Ø
              SA=VB(K)                                               L1  398Ø
              SB=GINV(K,I)                                           L1  399Ø
   67         S=S+SA*SB                                              L1  4ØØØ
            K=IR(I)                                                  L1  4Ø1Ø
   68       A(K)=S                                                   L1  4Ø2Ø
         S=Ø.                                                        L1  4Ø3Ø
         DO 69 J=1,N                                                 L1  4Ø4Ø
            SA=R(J)                                                  L1  4Ø5Ø
            IF(SA.LT.Ø.Ø) SA=-SA                                     L1  4Ø6Ø
   69       S=S+SA                                                   L1  4Ø7Ø
         Z=S                                                         L1  4Ø8Ø
         IF(IND.NE.Ø) RETURN                                         L1  4Ø9Ø
         E=2.-EPS                                                    L1  41ØØ
```

```
      DO 7Ø I=1,IRANK                                                  L1  411Ø
        D=XP(I)                                                        L1  412Ø
        IF(D.LT.EPS.OR.D.GT.E) IND=1                                   L1  413Ø
   7Ø   CONTINUE                                                       L1  414Ø
      RETURN                                                           L1  415Ø
      END                                                              L1  416Ø
```

```
 1.478689  1.113268  Ø.855Ø48  Ø.673991  Ø.541967
 Ø.442372  Ø.365257  Ø.3Ø4338  Ø.255453  Ø.215730
 1.Ø27547  Ø.884824  Ø.73ØØØ9  Ø.59849Ø  Ø.492769
 Ø.4Ø8461  Ø.34Ø883  Ø.286251  Ø.241692  Ø.2Ø5Ø51
 Ø.6926Ø8  Ø.652965  Ø.5775Ø5  Ø.496487  Ø.422211
 Ø.357993  Ø.3Ø3718  Ø.258Ø8  Ø.22Ø1Ø2  Ø.188148
 Ø.4435Ø3  Ø.452751  Ø.427474  Ø.386389  Ø.341112
 Ø.297441  Ø.25777Ø  Ø.222784  Ø.192393  Ø.166192
 Ø.251763  Ø.286357  Ø.291899  Ø.2795Ø5  Ø.257866
 Ø.232626  Ø.2Ø7Ø2Ø  Ø.182721  Ø.16Ø481  Ø.14Ø547
 Ø.1ØØ448  Ø.149Ø7Ø  Ø.173775  Ø.181324  Ø.17782Ø
 Ø.167924  Ø.154817  Ø.14Ø517  Ø.126221  Ø.112594
-Ø.Ø2Ø74Ø  Ø.Ø35871  Ø.Ø72667  Ø.Ø93931  Ø.1Ø3914
 Ø.1Ø6235  Ø.1Ø3674  Ø.Ø98224  Ø.Ø9124Ø  Ø.Ø836Ø7
-Ø.118431 -Ø.Ø57358 -Ø.Ø12925  Ø.Ø177Ø3  Ø.Ø37524
 Ø.Ø49293  Ø.Ø55315  Ø.Ø57387  Ø.Ø56853  Ø.Ø54674
-Ø.197141 -Ø.133831 -Ø.Ø84691 -Ø.Ø47765 -Ø.Ø2Ø895
-Ø.ØØ1986  Ø.Ø1Ø826  Ø.Ø19Ø96  Ø.Ø24Ø61  Ø.Ø26676
-Ø.26Ø1Ø2 -Ø.196Ø48 -Ø.144211 -Ø.1Ø3189 -Ø.Ø71395
-Ø.Ø47222 -Ø.Ø29174 -Ø.Ø15946 -Ø.ØØ6419  Ø.ØØØ282
       5.        3.        4.       12.        4.        7.
       9.        7.        3.       19.       13.        4.
       6.        6.        Ø.       12.       12.        2.
       9.        9.        7.       25.       11.        7.
       3.        Ø.        1.        4.        2.        7.
       8.        1.        8.       17.        1.        7.
       1.        9.        8.       18.        2.        3.
       Ø.        9.        3.       12.        6.        3.
       3.        1.        1.        5.        3.        5.
       6.        7.        6.       19.        7.        1.
       6.        1.        9.       16.       -2.        4.
       Ø.        4.        8.       12.       -4.        1.
       Ø.        5.        7.       12.       -2.        6.
       7.        3.        2.       12.        8.        6.
       5.        4.        9.       18.        Ø.        Ø.
```

```
      SUBROUTINE SUNDER                                                SUN ØØ1Ø
C     THIS IS A DUMMY SUBROUTINE. ITS FUNCTION IS TO                   SUN ØØ2Ø
C SUPPRESS THE UNDERFLOWS OCCURING IN THE CALCULATION.                 SUN ØØ3Ø
      RETURN                                                           SUN ØØ4Ø
      END                                                              SUN ØØ5Ø
```

```
      SUBROUTINE SETCLK                                                SET ØØ1Ø
C     THIS IS A DUMMY SUBROUTINE. ITS FUNCTION IS TO START             SET ØØ2Ø
C READING THE CPU TIME.                                                SET ØØ3Ø
      RETURN                                                           SET ØØ4Ø
      END                                                              SET ØØ5Ø
```

```
      SUBROUTINE RDCLK(TX)                                             RDC ØØ1Ø
C     THIS IS A DUMMY SUBROUTINE. ITS FUNCTION IS TO RECORD            RDC ØØ2Ø
C THE CPU TIME  TX  IN SECONDS, WHERE TX IS A REAL VARIABLE.           RDC ØØ3Ø
      TX=Ø.ØØ                                                          RDC ØØ4Ø
      RETURN                                                           RDC ØØ5Ø
      END                                                              RDC ØØ6Ø
```

ALGORITHM 552
Solution of the Constrained l_1 Linear Approximation Problem [F4]

I. BARRODALE and F. D. K. ROBERTS
University of Victoria, Canada

Key Words and Phrases: constrained l_1 approximation, linear programming, simplex method
CR Categories: 5.13, 5.41
Language: Fortran

DESCRIPTION

1. Introduction

Given a $k \times n$ system of linear equations

$$Ax = b, \tag{1}$$

the algorithm calculates an l_1 solution of (1) subject to l linear equality constraints

$$Cx = d \tag{2}$$

and m linear inequality constraints

$$Ex \leq f; \tag{3}$$

i.e., the subroutine determines a column vector x^* which minimizes

$$\| b - Ax \|_1 = \sum_{i=1}^{k} | b_i - A_i x | \tag{4}$$

subject to the given constraints (2) and (3). (In expression (4), b_i denotes the ith component of b and A_i denotes the ith row of A.)

The method is completely general in the sense that no restrictions are imposed on the ranks of the matrices A, C, and E, or on the signs of the elements of f. Furthermore, if no vector x satisfying (2) and (3) exists, the subroutine detects this and informs the user that the problem is infeasible.

The algorithm can be used to solve the constrained l_1 approximation problem. Suppose that data consisting of k points (t_i, y_i) are to be approximated by a linear approximating function $x_1\phi_1(t) + x_2\phi_2(t) + \cdots + x_n\phi_n(t)$, where certain linear constraints are imposed on the parameters $x_1, x_2, \ldots, x_n$. This is equivalent to finding an l_1 solution to the system of equations

$$\sum_{j=1}^{n} \phi_j(t_i)x_j = y_i, \qquad i = 1, 2, \ldots, k,$$

subject to the given linear constraints. If the data values y_i contain some large

Received 2 September 1977; revised 11 June 1979; accepted 24 July 1979.

Authors' address: Department of Mathematics, University of Victoria, P.O. Box 1700, Victoria, B. C., Canada V8W 2Y2.

errors, then an l_1 approximation may be preferable to an l_2 (least-squares) solution. This aspect of robust regression is discussed, for example, in [5, 7].

The algorithm is a modification of the simplex method applied to the primal formulation of the constrained l_1 problem as a linear program. The modification allows many intermediate simplex vertices to be bypassed, thus reducing the total number of simplex iterations required. A complete description of the method is given in [3]. In the absence of constraints, the algorithm reduces to our earlier unconstrained l_1 algorithm [1], which has proved to be most effective in practice. Indeed, in this case the program which follows produces identical results to the code given in [2].

Following [3], the constrained l_1 problem is posed as the following linear programming problem:

$$\begin{aligned} \text{minimize} \quad & e(u+v) + Me'(u'+v') + Me''v'' \\ \text{subject to} \quad & A(x'-x'') + u - v = b \\ & C(x'-x'') + u' - v' = d \\ & E(x'-x'') + u'' - v'' = f \\ & x', x'' \geq 0, \qquad u, v \geq 0, \qquad u', v' \geq 0, \qquad u'', v'' \geq 0. \end{aligned} \tag{5}$$

In (5), e, e', and e'' are vectors of 1's of appropriate dimensions and u', v', and v'' are artificial vectors with a large positive cost M in the objective function. Our modified simplex method is applied to a condensed tableau of size $(k + l + m + 2) \times (n + 2)$, using the usual two-phase approach in which the artificial part of the objective function is minimized first.

Our algorithm can be used to calculate a weighted constrained l_1 solution, merely by multiplying A_i and b_i by a chosen positive weight before entering the subroutine. Also, as is explained in the comment section of our listing, simple nonnegativity constraints on the variables need not be included explicitly in the contraints (3), since these can be handled most efficiently by assigning the appropriate variables x_j'' in (5) a cost M in the objective function, thus making them artificial variables. Similarly, a nonnegativity constraint on the ith row $b_i - A_i x$ can be handled by making the variable v_i in (5) artificial.

2. Numerical Results

The algorithm has been tested on a variety of problems, several of which are presented in [3]. We include here the following three additional test problems, which have been run on an IBM 370/158 using single-precision arithmetic (approximately 7 decimal digits).

Example 1. Bartels et al. [4] present an example of a spline approximation due to Maurice Cox which is constrained to be convex. This corresponds to an l_1 solution of a 9×7 system of equations whose unknowns are subject to 5 inequality constraints. Our algorithm requires 10 simplex iterations to reproduce (to single-precision accuracy) the results given in [4], where 17 iterations of their penalty function technique are required.

Example 2. This example was devised to demonstrate the ability of our algorithm to handle rank deficient problems. The matrices and vectors are defined as follows:

$$A = \begin{bmatrix} 2 & 0 & 1 & 3 & 1 \\ 7 & 4 & 4 & 15 & 7 \\ 9 & 4 & 7 & 20 & 6 \\ 2 & 2 & 1 & 5 & 3 \\ 9 & 3 & 2 & 14 & 10 \\ 4 & 5 & 0 & 9 & 9 \\ 4 & 4 & 9 & 17 & -1 \\ 1 & 6 & 2 & 9 & 5 \end{bmatrix}, \qquad b = \begin{bmatrix} 7 \\ 4 \\ 7 \\ 4 \\ 0 \\ 4 \\ 9 \\ 6 \end{bmatrix},$$

$$C = \begin{bmatrix} 0 & 4 & 5 & 9 & -1 \\ 3 & 2 & 7 & 12 & -2 \\ 3 & 6 & 12 & 21 & -3 \end{bmatrix}, \quad d = \begin{bmatrix} 5 \\ 1 \\ 6 \end{bmatrix},$$

$$E = \begin{bmatrix} 0 & 3 & 6 & 9 & -3 \\ 6 & 2 & 4 & 12 & 4 \end{bmatrix}, \quad f = \begin{bmatrix} 5 \\ 6 \end{bmatrix}.$$

The 8×5 matrix A is of rank 3. The 3×5 matrix C is of rank 2, which causes one artificial vector to remain in the basis at the end of phase I. Our algorithm includes an adaptation of a standard device (e.g., see Hadley [6, p. 153]) to handle such situations. The solution is obtained with no difficulty in 3 iterations with ERROR = 26.148.

Example 3. Compute a best l_1 approximation by $p_5(t) = \sum_{j=1}^{6} x_j t^{j-1}$ to the function $y(t) = \min(e^t, e^{1/2})$ on the 101 points defined by $t = 0(0.01)1$, subject to the interpolation condition that $p_5(t) = y(t)$ for $t = 0$, 0.5, and 1. This gives rise to a 98×6 overdetermined system with 3 equality constraints. The algorithm requires 8 iterations to solve the problem (with ERROR = 2.8513), and it requires 13 iterations to solve the 101×6 overdetermined system *without* the 3 constraints (with ERROR = 1.1473).

REFERENCES

1. BARRODALE, I., AND ROBERTS, F.D.K. An improved algorithm for discrete l_1 linear approximation. *SIAM J. Numer. Anal. 10* (1973), 839–848.
2. BARRODALE, I., AND ROBERTS, F.D.K. Algorithm 478: Solution of an overdetermined system of equations in the l_1 norm. *Commun. ACM 17* (1974), 319–320.
3. BARRODALE, I., AND ROBERTS F.D.K. An efficient algorithm for discrete l_1 linear approximation with linear constraints. *SIAM J. Numer. Anal. 15* (1978), 603–611.
4. BARTELS, R.H., CONN, A.R., AND SINCLAIR, J.W. Minimization techniques for piecewise differentiable functions: The l_1 solution to an overdetermined linear system. *SIAM J. Numer. Anal. 15* (1978), 224–241.
5. EKBLOM, H. L_p methods for robust regression. *BIT 14* (1974), 22–32.
6. HADLEY, G. *Linear Programming*. Addison-Wesley, Reading, Mass., 1962.
7. RICE, J.R., AND WHITE, J.S. Norms for smoothing and estimation. *SIAM Rev. 6* (1964), 243–256.

ALGORITHM

```
C SAMPLE DRIVER PROGRAM FOR SUBROUTINE CL1.                              MAN   2Ø
C                                                                        MAN   3Ø
C THIS PROGRAM SOLVES A K BY N OVERDETERMINED SYSTEM                     MAN   4Ø
C                                                                        MAN   5Ø
C    AX=B                                                                MAN   6Ø
C                                                                        MAN   7Ø
C IN THE L1 SENSE SUBJECT TO L EQUALITY CONSTRAINTS                      MAN   8Ø
C                                                                        MAN   9Ø
C    CX=D                                                                MAN  1ØØ
C                                                                        MAN  11Ø
C AND M INEQUALITY CONSTRAINTS                                           MAN  12Ø
C                                                                        MAN  13Ø
C    EX.LE.F.                                                            MAN  14Ø
C                                                                        MAN  15Ø
C COMPLETE DETAILS OF THE PARAMETERS MAY BE                              MAN  16Ø
C FOUND IN THE DOCUMENTATION OF THE SUBROUTINE.                          MAN  17Ø
C                                                                        MAN  18Ø
C THE ARRAYS ARE CURRENTLY DIMENSIONED TO ALLOW PROBLEMS                 MAN  19Ø
C FOR WHICH K+L+M .LE. 1ØØ, N .LE. 1Ø.                                  MAN  2ØØ
C                                                                        MAN  21Ø
C THE PROGRAM MAY BE TESTED ON THE FOLLOWING DATA.                       MAN  22Ø
C                                                                        MAN  23Ø
C     K = 8                                                              MAN  24Ø
C     L = 3                                                              MAN  25Ø
C     M = 2                                                              MAN  26Ø
C     N = 5                                                              MAN  27Ø
C                                                                        MAN  28Ø
C      Q = 2  Ø  1  3  1  7                                              MAN  29Ø
C          7  4  4 15  7  4                                              MAN  3ØØ
C          9  4  7 2Ø  6  7                                              MAN
```

```
C          2  2  1  5  3  4                                          MAN  31Ø
C          9  3  2 14 1Ø  Ø                                          MAN  32Ø
C          4  5  Ø  9  9  4                                          MAN  33Ø
C          4  4  9 17 -1  9                                          MAN  34Ø
C          1  6  2  9  5  6                                          MAN  35Ø
C          Ø  4  5  9 -1  5                                          MAN  36Ø
C          3  2  7 12 -2  1                                          MAN  37Ø
C          3  6 12 21 -3  6                                          MAN  38Ø
C          Ø  3  6  9 -3  5                                          MAN  39Ø
C          6  2  4 12  4  6                                          MAN  4ØØ
C                                                                    MAN  41Ø
C     KODE = Ø                                                       MAN  42Ø
C     TOLER = 1.E-5                                                  MAN  43Ø
C     ITER = 13Ø                                                     MAN  44Ø
C                                                                    MAN  45Ø
C                                                                    MAN  46Ø
      DIMENSION Q(1Ø2,12), X(12), RES(1ØØ), CU(2,11Ø)                MAN  47Ø
      INTEGER IU(2,11Ø), S(1ØØ)                                      MAN  48Ø
      DATA KLMD, KLM2D, NKLMD, N2D /1ØØ,1Ø2,11Ø,12/                  MAN  49Ø
C INPUT DATA.                                                        MAN  5ØØ
      READ (5,99999) K, L, M, N, KODE, TOLER, ITER                   MAN  51Ø
      KLM = K + L + M                                                MAN  52Ø
      N1 = N + 1                                                     MAN  53Ø
      DO 1Ø I=1,KLM                                                  MAN  54Ø
         READ (5,99998) (Q(I,J),J=1,N1)                              MAN  55Ø
         WRITE (6,99994) (Q(I,J),J=1,N1)                             MAN  56Ø
   1Ø CONTINUE                                                       MAN  57Ø
      CALL CL1(K, L, M, N, KLMD, KLM2D, NKLMD, N2D, Q,               MAN  58Ø
     * KODE, TOLER, ITER, X, RES, ERROR, CU, IU, S)                  MAN  59Ø
C OUTPUT KODE, ITERATION COUNT AND ERROR NORM.                       MAN  6ØØ
      WRITE (6,99997) KODE, ITER, ERROR                              MAN  61Ø
C OUTPUT SOLUTION VECTOR.                                            MAN  62Ø
      WRITE (6,99996) (I,X(I),I=1,N)                                 MAN  63Ø
C OUTPUT RESIDUAL ERROR AT EACH POINT.                               MAN  64Ø
      WRITE (6,99995) (I,RES(I),I=1,KLM)                             MAN  65Ø
      STOP                                                           MAN  66Ø
99999 FORMAT (5I3, E1Ø.Ø, I3)                                        MAN  67Ø
99998 FORMAT (8F3.Ø)                                                 MAN  68Ø
99997 FORMAT (16H KODE,ITER,ERROR, 2I1Ø, E18.7)                      MAN  69Ø
99996 FORMAT (4H SOL, I5, E18.7)                                     MAN  7ØØ
99995 FORMAT (6H ERROR, I5, E18.7)                                   MAN  71Ø
99994 FORMAT (2H  , 8F5.Ø)                                           MAN  72Ø
      END                                                            MAN  73Ø


      SUBROUTINE CL1(K, L, M, N, KLMD, KLM2D, NKLMD, N2D,            CL1   1Ø
     * Q, KODE, TOLER, ITER, X, RES, ERROR, CU, IU, S)               CL1   2Ø
C THIS SUBROUTINE USES A MODIFICATION OF THE SIMPLEX                 CL1   3Ø
C METHOD OF LINEAR PROGRAMMING TO CALCULATE AN L1 SOLUTION           CL1   4Ø
C TO A K BY N SYSTEM OF LINEAR EQUATIONS                             CL1   5Ø
C              AX=B                                                  CL1   6Ø
C SUBJECT TO L LINEAR EQUALITY CONSTRAINTS                           CL1   7Ø
C              CX=D                                                  CL1   8Ø
C AND M LINEAR INEQUALITY CONSTRAINTS                                CL1   9Ø
C              EX.LE.F.                                              CL1  1ØØ
C DESCRIPTION OF PARAMETERS                                          CL1  11Ø
C K       NUMBER OF ROWS OF THE MATRIX A (K.GE.1).                   CL1  12Ø
C L       NUMBER OF ROWS OF THE MATRIX C (L.GE.Ø).                   CL1  13Ø
C M       NUMBER OF ROWS OF THE MATRIX E (M.GE.Ø).                   CL1  14Ø
C N       NUMBER OF COLUMNS OF THE MATRICES A,C,E (N.GE.1).          CL1  15Ø
C KLMD    SET TO AT LEAST K+L+M FOR ADJUSTABLE DIMENSIONS.           CL1  16Ø
C KLM2D   SET TO AT LEAST K+L+M+2 FOR ADJUSTABLE DIMENSIONS.         CL1  17Ø
C NKLMD   SET TO AT LEAST N+K+L+M FOR ADJUSTABLE DIMENSIONS.         CL1  18Ø
C N2D     SET TO AT LEAST N+2 FOR ADJUSTABLE DIMENSIONS              CL1  19Ø
C Q       TWO DIMENSIONAL REAL ARRAY WITH KLM2D ROWS AND             CL1  2ØØ
C         AT LEAST N2D COLUMNS.                                      CL1  21Ø
C         ON ENTRY THE MATRICES A,C AND E, AND THE VECTORS           CL1  22Ø
C         B,D AND F MUST BE STORED IN THE FIRST K+L+M ROWS           CL1  23Ø
C         AND N+1 COLUMNS OF Q AS FOLLOWS                            CL1  24Ø
C              A B                                                   CL1  25Ø
C          Q = C D                                                   CL1  26Ø
C              E F                                                   CL1  27Ø
C         THESE VALUES ARE DESTROYED BY THE SUBROUTINE.              CL1  28Ø
C KODE    A CODE USED ON ENTRY TO, AND EXIT                          CL1  29Ø
C         FROM, THE SUBROUTINE.                                      CL1  3ØØ
C         ON ENTRY, THIS SHOULD NORMALLY BE SET TO Ø.                CL1  31Ø
```

```
C        HOWEVER, IF CERTAIN NONNEGATIVITY CONSTRAINTS                  CL1  32Ø
C        ARE TO BE INCLUDED IMPLICITLY, RATHER THAN                     CL1  33Ø
C        EXPLICITLY IN THE CONSTRAINTS EX.LE.F, THEN KODE               CL1  34Ø
C        SHOULD BE SET TO 1, AND THE NONNEGATIVITY                      CL1  35Ø
C        CONSTRAINTS INCLUDED IN THE ARRAYS X AND                       CL1  36Ø
C        RES (SEE BELOW).                                               CL1  37Ø
C        ON EXIT, KODE HAS ONE OF THE                                   CL1  38Ø
C        FOLLOWING VALUES                                               CL1  39Ø
C             Ø- OPTIMAL SOLUTION FOUND,                                CL1  4ØØ
C             1- NO FEASIBLE SOLUTION TO THE                            CL1  41Ø
C                CONSTRAINTS,                                           CL1  42Ø
C             2- CALCULATIONS TERMINATED                                CL1  43Ø
C                PREMATURELY DUE TO ROUNDING ERRORS,                    CL1  44Ø
C             3- MAXIMUM NUMBER OF ITERATIONS REACHED.                  CL1  45Ø
C TOLER  A SMALL POSITIVE TOLERANCE. EMPIRICAL                          CL1  46Ø
C        EVIDENCE SUGGESTS TOLER = 1Ø**(-D*2/3),                        CL1  47Ø
C        WHERE D REPRESENTS THE NUMBER OF DECIMAL                       CL1  48Ø
C        DIGITS OF ACCURACY AVAILABLE. ESSENTIALLY,                     CL1  49Ø
C        THE SUBROUTINE CANNOT DISTINGUISH BETWEEN ZERO                 CL1  5ØØ
C        AND ANY QUANTITY WHOSE MAGNITUDE DOES NOT EXCEED               CL1  51Ø
C        TOLER. IN PARTICULAR, IT WILL NOT PIVOT ON ANY                 CL1  52Ø
C        NUMBER WHOSE MAGNITUDE DOES NOT EXCEED TOLER.                  CL1  53Ø
C ITER   ON ENTRY ITER MUST CONTAIN AN UPPER BOUND ON                   CL1  54Ø
C        THE MAXIMUM NUMBER OF ITERATIONS ALLOWED.                      CL1  55Ø
C        A SUGGESTED VALUE IS 1Ø*(K+L+M). ON EXIT ITER                  CL1  56Ø
C        GIVES THE NUMBER OF SIMPLEX ITERATIONS.                        CL1  57Ø
C X      ONE DIMENSIONAL REAL ARRAY OF SIZE AT LEAST N2D.               CL1  58Ø
C        ON EXIT THIS ARRAY CONTAINS A                                  CL1  59Ø
C        SOLUTION TO THE L1 PROBLEM. IF KODE=1                          CL1  6ØØ
C        ON ENTRY, THIS ARRAY IS ALSO USED TO INCLUDE                   CL1  61Ø
C        SIMPLE NONNEGATIVITY CONSTRAINTS ON THE                        CL1  62Ø
C        VARIABLES. THE VALUES -1, Ø, OR 1                              CL1  63Ø
C        FOR X(J) INDICATE THAT THE J-TH VARIABLE                       CL1  64Ø
C        IS RESTRICTED TO BE .LE.Ø, UNRESTRICTED,                       CL1  65Ø
C        OR .GE.Ø RESPECTIVELY.                                         CL1  66Ø
C RES    ONE DIMENSIONAL REAL ARRAY OF SIZE AT LEAST KLMD.              CL1  67Ø
C        ON EXIT THIS CONTAINS THE RESIDUALS B-AX                       CL1  68Ø
C        IN THE FIRST K COMPONENTS, D-CX IN THE                         CL1  69Ø
C        NEXT L COMPONENTS (THESE WILL BE =Ø),AND                       CL1  7ØØ
C        F-EX IN THE NEXT M COMPONENTS. IF KODE=1 ON                    CL1  71Ø
C        ENTRY, THIS ARRAY IS ALSO USED TO INCLUDE SIMPLE               CL1  72Ø
C        NONNEGATIVITY CONSTRAINTS ON THE RESIDUALS                     CL1  73Ø
C        B-AX. THE VALUES -1, Ø, OR 1 FOR RES(I)                        CL1  74Ø
C        INDICATE THAT THE I-TH RESIDUAL (1.LE.I.LE.K) IS               CL1  75Ø
C        RESTRICTED TO BE .LE.Ø, UNRESTRICTED, OR .GE.Ø                 CL1  76Ø
C        RESPECTIVELY.                                                  CL1  77Ø
C ERROR  ON EXIT, THIS GIVES THE MINIMUM SUM OF                         CL1  78Ø
C        ABSOLUTE VALUES OF THE RESIDUALS.                              CL1  79Ø
C CU     A TWO DIMENSIONAL REAL ARRAY WITH TWO ROWS AND                 CL1  8ØØ
C        AT LEAST NKLMD COLUMNS USED FOR WORKSPACE.                     CL1  81Ø
C IU     A TWO DIMENSIONAL INTEGER ARRAY WITH TWO ROWS AND              CL1  82Ø
C        AT LEAST NKLMD COLUMNS USED FOR WORKSPACE.                     CL1  83Ø
C S      INTEGER ARRAY OF SIZE AT LEAST KLMD, USED FOR                  CL1  84Ø
C        WORKSPACE.                                                     CL1  85Ø
C IF YOUR FORTRAN COMPILER PERMITS A SINGLE COLUMN OF A TWO             CL1  86Ø
C DIMENSIONAL ARRAY TO BE PASSED TO A ONE DIMENSIONAL ARRAY             CL1  87Ø
C THROUGH A SUBROUTINE CALL, CONSIDERABLE SAVINGS IN                    CL1  88Ø
C EXECUTION TIME MAY BE ACHIEVED THROUGH THE USE OF THE                 CL1  89Ø
C FOLLOWING SUBROUTINE, WHICH OPERATES ON COLUMN VECTORS.               CL1  9ØØ
C     SUBROUTINE COL(V1, V2, XMLT, NOTROW, K)                           CL1  91Ø
C THIS SUBROUTINE ADDS TO THE VECTOR V1 A MULTIPLE OF THE               CL1  92Ø
C VECTOR V2 (ELEMENTS 1 THROUGH K EXCLUDING NOTROW).                    CL1  93Ø
C     DIMENSION V1(K), V2(K)                                            CL1  94Ø
C     KEND = NOTROW - 1                                                 CL1  95Ø
C     KSTART = NOTROW + 1                                               CL1  96Ø
C     IF (KEND .LT. 1) GO TO 2Ø                                         CL1  97Ø
C     DO 1Ø I=1,KEND                                                    CL1  98Ø
C        V1(I) = V1(I) + XMLT*V2(I)                                     CL1  99Ø
C  1Ø CONTINUE                                                          CL1 1ØØØ
C     IF(KSTART .GT. K) GO TO 4Ø                                        CL1 1Ø1Ø
C  2Ø DO 3Ø I=KSTART,K                                                  CL1 1Ø2Ø
C       V1(I) = V1(I) + XMLT*V2(I)                                      CL1 1Ø3Ø
C  3Ø CONTINUE                                                          CL1 1Ø4Ø
C  4Ø RETURN                                                            CL1 1Ø5Ø
C     END                                                               CL1 1Ø6Ø
C SEE COMMENTS FOLLOWING STATEMENT LABELLED 44Ø FOR                     CL1 1Ø7Ø
```

```
C INSTRUCTIONS ON THE IMPLEMENTATION OF THIS MODIFICATION.          CL1 1Ø8Ø
      DOUBLE PRECISION SUM                                         CL1 1Ø9Ø
      DOUBLE PRECISION DBLE                                        CL1 11ØØ
      REAL Q, X, Z, CU, SN, ZU, ZV, CUV, RES, XMAX, XMIN,          CL1 111Ø
     * ERROR, PIVOT, TOLER, TPIVOT                                 CL1 112Ø
      REAL ABS                                                     CL1 113Ø
      INTEGER I, J, K, L, M, N, S, IA, II, IN, IU, JS, KK,         CL1 114Ø
     * NK, N1, N2, JMN, JPN, KLM, NKL, NK1, N2D, IIMN,            CL1 115Ø
     * IOUT, ITER, KLMD, KLM1, KLM2, KODE, NKLM, NKL1,            CL1 116Ø
     * KLM2D, MAXIT, NKLMD, IPHASE, KFORCE, IINEG                 CL1 117Ø
      INTEGER IABS                                                 CL1 118Ø
      DIMENSION Q(KLM2D,N2D), X(N2D), RES(KLMD),                   CL1 119Ø
     * CU(2,NKLMD), IU(2,NKLMD), S(KLMD)                          CL1 12ØØ
C                                                                  CL1 121Ø
C INITIALIZATION.                                                  CL1 122Ø
C                                                                  CL1 123Ø
      MAXIT = ITER                                                 CL1 124Ø
      N1 = N + 1                                                   CL1 125Ø
      N2 = N + 2                                                   CL1 126Ø
      NK = N + K                                                   CL1 127Ø
      NK1 = NK + 1                                                 CL1 128Ø
      NKL = NK + L                                                 CL1 129Ø
      NKL1 = NKL + 1                                               CL1 13ØØ
      KLM = K + L + M                                              CL1 131Ø
      KLM1 = KLM + 1                                               CL1 132Ø
      KLM2 = KLM + 2                                               CL1 133Ø
      NKLM = N + KLM                                               CL1 134Ø
      KFORCE = 1                                                   CL1 135Ø
      ITER = Ø                                                     CL1 136Ø
      JS = 1                                                       CL1 137Ø
      IA = Ø                                                       CL1 138Ø
C SET UP LABELS IN Q.                                              CL1 139Ø
      DO 1Ø J=1,N                                                  CL1 14ØØ
         Q(KLM2,J) = J                                             CL1 141Ø
   1Ø CONTINUE                                                     CL1 142Ø
      DO 3Ø I=1,KLM                                                CL1 143Ø
         Q(I,N2) = N + I                                           CL1 144Ø
         IF (Q(I,N1).GE.Ø.) GO TO 3Ø                               CL1 145Ø
         DO 2Ø J=1,N2                                              CL1 146Ø
            Q(I,J) = -Q(I,J)                                       CL1 147Ø
   2Ø    CONTINUE                                                  CL1 148Ø
   3Ø CONTINUE                                                     CL1 149Ø
C SET UP PHASE 1 COSTS.                                            CL1 15ØØ
      IPHASE = 2                                                   CL1 151Ø
      DO 4Ø J=1,NKLM                                               CL1 152Ø
         CU(1,J) = Ø.                                              CL1 153Ø
         CU(2,J) = Ø.                                              CL1 154Ø
         IU(1,J) = Ø                                               CL1 155Ø
         IU(2,J) = Ø                                               CL1 156Ø
   4Ø CONTINUE                                                     CL1 157Ø
      IF (L.EQ.Ø) GO TO 6Ø                                         CL1 158Ø
      DO 5Ø J=NK1,NKL                                              CL1 159Ø
         CU(1,J) = 1.                                              CL1 16ØØ
         CU(2,J) = 1.                                              CL1 161Ø
         IU(1,J) = 1                                               CL1 162Ø
         IU(2,J) = 1                                               CL1 163Ø
   5Ø CONTINUE                                                     CL1 164Ø
      IPHASE = 1                                                   CL1 165Ø
   6Ø IF (M.EQ.Ø) GO TO 8Ø                                         CL1 166Ø
      DO 7Ø J=NKL1,NKLM                                            CL1 167Ø
         CU(2,J) = 1.                                              CL1 168Ø
         IU(2,J) = 1                                               CL1 169Ø
         JMN = J - N                                               CL1 17ØØ
         IF (Q(JMN,N2).LT.Ø.) IPHASE = 1                           CL1 171Ø
   7Ø CONTINUE                                                     CL1 172Ø
   8Ø IF (KODE.EQ.Ø) GO TO 15Ø                                     CL1 173Ø
      DO 11Ø J=1,N                                                 CL1 174Ø
         IF (X(J)) 9Ø, 11Ø, 1ØØ                                    CL1 175Ø
   9Ø    CU(1,J) = 1.                                              CL1 176Ø
         IU(1,J) = 1                                               CL1 177Ø
         GO TO 11Ø                                                 CL1 178Ø
  1ØØ    CU(2,J) = 1.                                              CL1 179Ø
         IU(2,J) = 1                                               CL1 18ØØ
  11Ø CONTINUE                                                     CL1 181Ø
      DO 14Ø J=1,K                                                 CL1 182Ø
         JPN = J + N                                               CL1 183Ø
```

```
            IF (RES(J)) 12Ø, 14Ø, 13Ø                                   CL1 184Ø
   12Ø      CU(1,JPN) = 1.                                              CL1 185Ø
            IU(1,JPN) = 1                                               CL1 186Ø
            IF (Q(J,N2).GT.Ø.Ø) IPHASE = 1                              CL1 187Ø
            GO TO 14Ø                                                   CL1 188Ø
   13Ø      CU(2,JPN) = 1.                                              CL1 189Ø
            IU(2,JPN) = 1                                               CL1 19ØØ
            IF (Q(J,N2).LT.Ø.Ø) IPHASE = 1                              CL1 191Ø
   14Ø CONTINUE                                                         CL1 192Ø
   15Ø IF (IPHASE.EQ.2) GO TO 5ØØ                                       CL1 193Ø
C  COMPUTE THE MARGINAL COSTS.                                          CL1 194Ø
   16Ø DO 2ØØ J=JS,N1                                                   CL1 195Ø
            SUM = Ø.DØ                                                  CL1 196Ø
            DO 19Ø I=1,KLM                                              CL1 197Ø
               II = Q(I,N2)                                             CL1 198Ø
               IF (II.LT.Ø) GO TO 17Ø                                   CL1 199Ø
               Z = CU(1,II)                                             CL1 2ØØØ
               GO TO 18Ø                                                CL1 2Ø1Ø
   17Ø         IINEG = -II                                              CL1 2Ø2Ø
               Z = CU(2,IINEG)                                          CL1 2Ø3Ø
   18Ø         SUM = SUM + DBLE(Q(I,J))*DBLE(Z)                         CL1 2Ø4Ø
   19Ø      CONTINUE                                                    CL1 2Ø5Ø
            Q(KLM1,J) = SUM                                             CL1 2Ø6Ø
   2ØØ CONTINUE                                                         CL1 2Ø7Ø
       DO 23Ø J=JS,N                                                    CL1 2Ø8Ø
            II = Q(KLM2,J)                                              CL1 2Ø9Ø
            IF (II.LT.Ø) GO TO 21Ø                                      CL1 21ØØ
            Z = CU(1,II)                                                CL1 211Ø
            GO TO 22Ø                                                   CL1 212Ø
   21Ø      IINEG = -II                                                 CL1 213Ø
            Z = CU(2,IINEG)                                             CL1 214Ø
   22Ø      Q(KLM1,J) = Q(KLM1,J) - Z                                   CL1 215Ø
   23Ø CONTINUE                                                         CL1 216Ø
C  DETERMINE THE VECTOR TO ENTER THE BASIS.                             CL1 217Ø
   24Ø XMAX = Ø.                                                        CL1 218Ø
       IF (JS.GT.N) GO TO 49Ø                                           CL1 219Ø
       DO 28Ø J=JS,N                                                    CL1 22ØØ
            ZU = Q(KLM1,J)                                              CL1 221Ø
            II = Q(KLM2,J)                                              CL1 222Ø
            IF (II.GT.Ø) GO TO 25Ø                                      CL1 223Ø
            II = -II                                                    CL1 224Ø
            ZV = ZU                                                     CL1 225Ø
            ZU = -ZU - CU(1,II) - CU(2,II)                              CL1 226Ø
            GO TO 26Ø                                                   CL1 227Ø
   25Ø      ZV = -ZU - CU(1,II) - CU(2,II)                              CL1 228Ø
   26Ø      IF (KFORCE.EQ.1 .AND. II.GT.N) GO TO 28Ø                    CL1 229Ø
            IF (IU(1,II).EQ.1) GO TO 27Ø                                CL1 23ØØ
            IF (ZU.LE.XMAX) GO TO 27Ø                                   CL1 231Ø
            XMAX = ZU                                                   CL1 232Ø
            IN = J                                                      CL1 233Ø
   27Ø      IF (IU(2,II).EQ.1) GO TO 28Ø                                CL1 234Ø
            IF (ZV.LE.XMAX) GO TO 28Ø                                   CL1 235Ø
            XMAX = ZV                                                   CL1 236Ø
            IN = J                                                      CL1 237Ø
   28Ø CONTINUE                                                         CL1 238Ø
       IF (XMAX.LE.TOLER) GO TO 49Ø                                     CL1 239Ø
       IF (Q(KLM1,IN).EQ.XMAX) GO TO 3ØØ                                CL1 24ØØ
       DO 29Ø I=1,KLM2                                                  CL1 241Ø
            Q(I,IN) = -Q(I,IN)                                          CL1 242Ø
   29Ø CONTINUE                                                         CL1 243Ø
       Q(KLM1,IN) = XMAX                                                CL1 244Ø
C  DETERMINE THE VECTOR TO LEAVE THE BASIS.                             CL1 245Ø
   3ØØ IF (IPHASE.EQ.1 .OR. IA.EQ.Ø) GO TO 33Ø                          CL1 246Ø
       XMAX = Ø.                                                        CL1 247Ø
       DO 31Ø I=1,IA                                                    CL1 248Ø
            Z = ABS(Q(I,IN))                                            CL1 249Ø
            IF (Z.LE.XMAX) GO TO 31Ø                                    CL1 25ØØ
            XMAX = Z                                                    CL1 251Ø
            IOUT = I                                                    CL1 252Ø
   31Ø CONTINUE                                                         CL1 253Ø
       IF (XMAX.LE.TOLER) GO TO 33Ø                                     CL1 254Ø
       DO 32Ø J=1,N2                                                    CL1 255Ø
            Z = Q(IA,J)                                                 CL1 256Ø
            Q(IA,J) = Q(IOUT,J)                                         CL1 257Ø
            Q(IOUT,J) = Z                                               CL1 258Ø
   32Ø CONTINUE                                                         CL1 259Ø
```

```
      IOUT = IA                                                    CL1 26ØØ
      IA = IA - 1                                                  CL1 261Ø
      PIVOT = Q(IOUT,IN)                                           CL1 262Ø
      GO TO 42Ø                                                    CL1 263Ø
  33Ø KK = Ø                                                       CL1 264Ø
      DO 34Ø I=1,KLM                                               CL1 265Ø
         Z = Q(I,IN)                                               CL1 266Ø
         IF (Z.LE.TOLER) GO TO 34Ø                                 CL1 267Ø
         KK = KK + 1                                               CL1 268Ø
         RES(KK) = Q(I,N1)/Z                                       CL1 269Ø
         S(KK) = I                                                 CL1 27ØØ
  34Ø CONTINUE                                                     CL1 271Ø
  35Ø IF (KK.GT.Ø) GO TO 36Ø                                       CL1 272Ø
      KODE = 2                                                     CL1 273Ø
      GO TO 59Ø                                                    CL1 274Ø
  36Ø XMIN = RES(1)                                                CL1 275Ø
      IOUT = S(1)                                                  CL1 276Ø
      J = 1                                                        CL1 277Ø
      IF (KK.EQ.1) GO TO 38Ø                                       CL1 278Ø
      DO 37Ø I=2,KK                                                CL1 279Ø
         IF (RES(I).GE.XMIN) GO TO 37Ø                             CL1 28ØØ
         J = I                                                     CL1 281Ø
         XMIN = RES(I)                                             CL1 282Ø
         IOUT = S(I)                                               CL1 283Ø
  37Ø CONTINUE                                                     CL1 284Ø
      RES(J) = RES(KK)                                             CL1 285Ø
      S(J) = S(KK)                                                 CL1 286Ø
  38Ø KK = KK - 1                                                  CL1 287Ø
      PIVOT = Q(IOUT,IN)                                           CL1 288Ø
      II = Q(IOUT,N2)                                              CL1 289Ø
      IF (IPHASE.EQ.1) GO TO 4ØØ                                   CL1 29ØØ
      IF (II.LT.Ø) GO TO 39Ø                                       CL1 291Ø
      IF (IU(2,II).EQ.1) GO TO 42Ø                                 CL1 292Ø
      GO TO 4ØØ                                                    CL1 293Ø
  39Ø IINEG = -II                                                  CL1 294Ø
      IF (IU(1,IINEG).EQ.1) GO TO 42Ø                              CL1 295Ø
  4ØØ II = IABS(II)                                                CL1 296Ø
      CUV = CU(1,II) + CU(2,II)                                    CL1 297Ø
      IF (Q(KLM1,IN)-PIVOT*CUV.LE.TOLER) GO TO 42Ø                 CL1 298Ø
C BYPASS INTERMEDIATE VERTICES.                                    CL1 299Ø
      DO 41Ø J=JS,N1                                               CL1 3ØØØ
         Z = Q(IOUT,J)                                             CL1 3Ø1Ø
         Q(KLM1,J) = Q(KLM1,J) - Z*CUV                             CL1 3Ø2Ø
         Q(IOUT,J) = -Z                                            CL1 3Ø3Ø
  41Ø CONTINUE                                                     CL1 3Ø4Ø
      Q(IOUT,N2) = -Q(IOUT,N2)                                     CL1 3Ø5Ø
      GO TO 35Ø                                                    CL1 3Ø6Ø
C GAUSS-JORDAN ELIMINATION.                                        CL1 3Ø7Ø
  42Ø IF (ITER.LT.MAXIT) GO TO 43Ø                                 CL1 3Ø8Ø
      KODE = 3                                                     CL1 3Ø9Ø
      GO TO 59Ø                                                    CL1 31ØØ
  43Ø ITER = ITER + 1                                              CL1 311Ø
      DO 44Ø J=JS,N1                                               CL1 312Ø
         IF (J.NE.IN) Q(IOUT,J) = Q(IOUT,J)/PIVOT                  CL1 313Ø
  44Ø CONTINUE                                                     CL1 314Ø
C IF PERMITTED, USE SUBROUTINE COL OF THE DESCRIPTION              CL1 315Ø
C SECTION AND REPLACE THE FOLLOWING SEVEN STATEMENTS DOWN          CL1 316Ø
C TO AND INCLUDING STATEMENT NUMBER 46Ø BY..                       CL1 317Ø
C     DO 46Ø J=JS,N1                                               CL1 318Ø
C        IF(J .EQ. IN) GO TO 46Ø                                   CL1 319Ø
C        Z = -Q(IOUT,J)                                            CL1 32ØØ
C        CALL COL(Q(1,J), Q(1,IN), Z, IOUT, KLM1)                  CL1 321Ø
C 46Ø CONTINUE                                                     CL1 322Ø
      DO 46Ø J=JS,N1                                               CL1 323Ø
         IF (J.EQ.IN) GO TO 46Ø                                    CL1 324Ø
         Z = -Q(IOUT,J)                                            CL1 325Ø
         DO 45Ø I=1,KLM1                                           CL1 326Ø
            IF (I.NE.IOUT) Q(I,J) = Q(I,J) + Z*Q(I,IN)             CL1 327Ø
  45Ø    CONTINUE                                                  CL1 328Ø
  46Ø CONTINUE                                                     CL1 329Ø
      TPIVOT = -PIVOT                                              CL1 33ØØ
      DO 47Ø I=1,KLM1                                              CL1 331Ø
         IF (I.NE.IOUT) Q(I,IN) = Q(I,IN)/TPIVOT                   CL1 332Ø
  47Ø CONTINUE                                                     CL1 333Ø
      Q(IOUT,IN) = 1./PIVOT                                        CL1 334Ø
      Z = Q(IOUT,N2)                                               CL1 335Ø
```

```
      Q(IOUT,N2) = Q(KLM2,IN)                                          CL1 336Ø
      Q(KLM2,IN) = Z                                                   CL1 337Ø
      II = ABS(Z)                                                      CL1 338Ø
      IF (IU(1,II).EQ.Ø .OR. IU(2,II).EQ.Ø) GO TO 24Ø                  CL1 339Ø
      DO 48Ø I=1,KLM2                                                  CL1 34ØØ
         Z = Q(I,IN)                                                   CL1 341Ø
         Q(I,IN) = Q(I,JS)                                             CL1 342Ø
         Q(I,JS) = Z                                                   CL1 343Ø
  48Ø CONTINUE                                                         CL1 344Ø
      JS = JS + 1                                                      CL1 345Ø
      GO TO 24Ø                                                        CL1 346Ø
C TEST FOR OPTIMALITY.                                                 CL1 347Ø
  49Ø IF (KFORCE.EQ.Ø) GO TO 58Ø                                       CL1 348Ø
      IF (IPHASE.EQ.1 .AND. Q(KLM1,N1).LE.TOLER) GO TO 5ØØ             CL1 349Ø
      KFORCE = Ø                                                       CL1 35ØØ
      GO TO 24Ø                                                        CL1 351Ø
C SET UP PHASE 2 COSTS.                                                CL1 352Ø
  5ØØ IPHASE = 2                                                       CL1 353Ø
      DO 51Ø J=1,NKLM                                                  CL1 354Ø
         CU(1,J) = Ø.                                                  CL1 355Ø
         CU(2,J) = Ø.                                                  CL1 356Ø
  51Ø CONTINUE                                                         CL1 357Ø
      DO 52Ø J=N1,NK                                                   CL1 358Ø
         CU(1,J) = 1.                                                  CL1 359Ø
         CU(2,J) = 1.                                                  CL1 36ØØ
  52Ø CONTINUE                                                         CL1 361Ø
      DO 56Ø I=1,KLM                                                   CL1 362Ø
         II = Q(I,N2)                                                  CL1 363Ø
         IF (II.GT.Ø) GO TO 53Ø                                        CL1 364Ø
         II = -II                                                      CL1 365Ø
         IF (IU(2,II).EQ.Ø) GO TO 56Ø                                  CL1 366Ø
         CU(2,II) = Ø.                                                 CL1 367Ø
         GO TO 54Ø                                                     CL1 368Ø
  53Ø    IF (IU(1,II).EQ.Ø) GO TO 56Ø                                  CL1 369Ø
         CU(1,II) = Ø.                                                 CL1 37ØØ
  54Ø    IA = IA + 1                                                   CL1 371Ø
         DO 55Ø J=1,N2                                                 CL1 372Ø
            Z = Q(IA,J)                                                CL1 373Ø
            Q(IA,J) = Q(I,J)                                           CL1 374Ø
            Q(I,J) = Z                                                 CL1 375Ø
  55Ø    CONTINUE                                                      CL1 376Ø
  56Ø CONTINUE                                                         CL1 377Ø
      GO TO 16Ø                                                        CL1 378Ø
  57Ø IF (Q(KLM1,N1).LE.TOLER) GO TO 5ØØ                               CL1 379Ø
      KODE = 1                                                         CL1 38ØØ
      GO TO 59Ø                                                        CL1 381Ø
  58Ø IF (IPHASE.EQ.1) GO TO 57Ø                                       CL1 382Ø
C PREPARE OUTPUT.                                                      CL1 383Ø
      KODE = Ø                                                         CL1 384Ø
  59Ø SUM = Ø.DØ                                                       CL1 385Ø
      DO 6ØØ J=1,N                                                     CL1 386Ø
         X(J) = Ø.                                                     CL1 387Ø
  6ØØ CONTINUE                                                         CL1 388Ø
      DO 61Ø I=1,KLM                                                   CL1 389Ø
         RES(I) = Ø.                                                   CL1 39ØØ
  61Ø CONTINUE                                                         CL1 391Ø
      DO 64Ø I=1,KLM                                                   CL1 392Ø
         II = Q(I,N2)                                                  CL1 393Ø
         SN = 1.                                                       CL1 394Ø
         IF (II.GT.Ø) GO TO 62Ø                                        CL1 395Ø
         II = -II                                                      CL1 396Ø
         SN = -1.                                                      CL1 397Ø
  62Ø    IF (II.GT.N) GO TO 63Ø                                        CL1 398Ø
         X(II) = SN*Q(I,N1)                                            CL1 399Ø
         GO TO 64Ø                                                     CL1 4ØØØ
  63Ø    IIMN = II - N                                                 CL1 4Ø1Ø
         RES(IIMN) = SN*Q(I,N1)                                        CL1 4Ø2Ø
         IF (II.GE.N1 .AND. II.LE.NK) SUM = SUM +                      CL1 4Ø3Ø
     *    DBLE(Q(I,N1))                                               CL1 4Ø4Ø
  64Ø CONTINUE                                                         CL1 4Ø5Ø
      ERROR = SUM                                                      CL1 4Ø6Ø
      RETURN                                                           CL1 4Ø7Ø
      END                                                              CL1 4Ø8Ø
```

```
 8  3  2  5  Ø      1.E-513Ø
 2  Ø  1  3  1  7
 7  4  4 15  7  4
 9  4  7 2Ø  6  7
 2  2  1  5  3  4
 9  3  2 14 1Ø  Ø
 4  5  Ø  9  9  4
 4  4  9 17 -1  9
 1  6  2  9  5  6
 Ø  4  5  9 -1  5
 3  2  7 12 -2  1
 3  6 12 21 -3  6
 Ø  3  6  9 -3  5
 6  2  4 12  4  6
```

ALGORITHM 553

M3RK, An Explicit Time Integrator for Semidiscrete Parabolic Equations [D3]

J. G. VERWER

Mathematical Center, Amsterdam, The Netherlands

Key Words and Phrases: parabolic partial differential equations, semidiscretization, explicit time integrator
CR Categories: 5.17
Language: Fortran

DESCRIPTION

This algorithm is a complement to [2] where it is explained and described.

REFERENCES

1. RYDER, B.G. The PFORT verifier. *Softw. Pract. Exper. 4* (1974), 359–378.
2. VERWER, J.G. An implementation of a class of stabilized explicit methods for the time integration of parabolic equations. *ACM Trans. Math. Softw. 6*, 2 (June 1980), 188–205.

ALGORITHM

```
C      PROGRAM TEST1(OUTPUT,TAPE6=OUTPUT)
C
C TEST PROGRAM 1 FOR J.G.VERWER'S ALGORITHM M3RK, AN EXPLICIT TIME-
C INTEGRATOR FOR SEMI-DISCRETE PARABOLIC EQUATIONS.
C THE TEST PROBLEM IS A SYSTEM OF 2 ONE-DIMENSIONAL PARABOLIC EQUA-
C TIONS WHICH IS DISCUSSED IN SECTION 5.1 OF J.G.VERWER'S COMPANION
C PAPER "AN IMPLEMENTATION OF A CLASS OF STABILIZED, EXPLICIT ME-
C THODS FOR THE TIME INTEGRATION OF PARABOLIC EQUTIONS".
C THE TEST CONSISTS OF THE INTEGRATION OF THE SYSTEM OF 122 ODE'S
C WHICH IS CALLED SYSTEM II IN SECTION 5.1 OF THE COMPANION PAPER
C (EQUATIONS (5.3) WITH 61 GRID POINTS FOR THE GALERKIN DISCRETI-
C ZATION). IN THE PRESENT TEST THE TOLERANCE PARAMETER TOL=1.ØE-4.
C
C ***************************************************************
C
C ARRAY DECLARATIONS FOR M3RK. THE ARRAY U IS THE INPUT ARRAY FOR
C THE INITIAL VECTOR AND UOUT IS THE OUTPUT ARRAY FOR THE COMPUTED
C SOLUTION VECTOR.
C DER IS THE SUBROUTINE DEFINING THE SYSTEM OF ODE'S.
C THE ARRAY X WILL CONTAIN THE GRID POINTS FOR THE GALERKIN DISCRE-
C TIZATION AND IS USED BY DER.
C
      DIMENSION INFO(15),U(122),U1(122),U2(122),UOUT(122),DU(122)
     1,DU1(122),SIGMA(2)
      EXTERNAL DER
      COMMON /GRID/ X(61)
C
C DEFINITION AND PRINTING OF THE GRID POINTS X(I).THE NUMBER OF
```

Received 22 June 1977; revised 25 July 1979; accepted 11 December 1979.

Author's address: Mathematisch Centrum, 2e Boerhaavestraat 49, 1091 AL Amsterdam, The Netherlands.

```
C GRID POINTS CAN BE CHANGED BY REDEFINING NPTS AND BY ADJUSTING
C THE LENGTH OT THE ARRAYS.
C
      NPTS=61
      DO 1Ø I=1,NPTS
   1Ø X(I)=(FLOAT(I)-1.)/(FLOAT(NPTS)-1.)
      WRITE(6,12)
   12 FORMAT(7H1 GRID=)
      DO 13 I=1,NPTS
   13 WRITE(6,14)X(I)
   14 FORMAT(F12.5)
C
C DEFINITION AND PRINTING OF THE INITIAL VECTOR FOR THE ODE'S.
C
      WRITE(6,2Ø)
   2Ø FORMAT(17H1 INITIAL VALUES=)
      WRITE(6,55)
      DO 3Ø I = 1,NPTS
      U(I) = 1.Ø
      IN=I+NPTS
      U(IN) = .Ø
      WRITE(6,7Ø)X(I),U(I),U(IN)
   3Ø CONTINUE
C
C DEFINITION OF PARAMETERS FOR M3RK. NEQN GIVES THE NUMBER OF COM-
C PONENTS OF THE SYSTEM OF ODE'S. T IS THE INDEPENDENT VARIABLE.
C THE MEANING OF THE OTHER PARAMETERS IS CLEARLY EXPLAINED IN THE
C PROLOGUE OF COMMENTS OF M3RK.
C
      NEQN=NPTS*2
      T=.Ø
      TOL=1.E-4
      INFO(1)=Ø
      INFO(2)=2
      INFO(3)=5ØØØ
C
C INITIAL AND SUBSEQUENT CALLS OF M3RK(SEE SECTION 5.1 OF THE
C COMPANION PAPER). TE DEFINES THE POINT WHERE OUTPUT IS DESIRED.
C
      DO 8Ø I=1,6
      TE=.Ø1
      IF(I.EQ.2)TE=.1
      IF(I.EQ.3)TE=1.
      IF(I.EQ.4)TE=5.
      IF(I.EQ.5)TE=1Ø.
      IF(I.EQ.6)TE=2Ø.
      CALL M3RK(T,TE,NEQN,H,HMIN,SIGMA,TOL,
     +          DER,U,U1,U2,UOUT,DU,DU1,IFLAG,INFO)
C
C PRINTING OF THE OUTPUT POINT, OF THE ERROR FLAG, OF THE ESTIMATED
C SPECTRAL RADIUS, OF THE ARRAY INFO, AND OF THE COMPUTED SOLUTION.
C
      WRITE(6,4Ø)TE,IFLAG,SIGMA(1)
   4Ø FORMAT(14H1 ENDPOINT   =,E1Ø.4/1ØH  IFLAG=  ,I2/8H  SIGMA=
     1,E13.7)
      WRITE(6,5Ø)(INFO(KK),KK=1,15)
   5Ø FORMAT(8H  INFO= ,15I6/)
      WRITE(6,55)
   55 FORMAT(5X,1HX,17X,1HU,19X,1HV/)
      DO 6Ø J=1,NPTS
      JN=J+NPTS
   6Ø WRITE(6,7Ø)X(J),UOUT(J),UOUT(JN)
   7Ø FORMAT(F9.5,2F2Ø.7)
   8Ø CONTINUE
      STOP
      END


      SUBROUTINE DER(N,Y)
C
C SUBROUTINE DER DEFINES THE SYSTEM OF ODE'S BEING INTEGRATED.
C THE ARRAY U IS A WORK ARRAY.
C
      DIMENSION Y(N),U(122)
      COMMON /GRID/ X(61)
      FI(Z)=EXP(Z*5.73)-EXP(-11.46*Z)
```

```
      DO 1Ø I=1,N
   1Ø U(I)=Y(I)
      EPS=Ø.143
      P=Ø.1743
      Y(1)=-2.*EPS*P*(7.*U(1)-8.*U(2)+U(3))/(X(3)-X(1))**2-FI(U(1)-U(62)
     1)
      Y(61)=.Ø
      Y(62)=.Ø
      Y(122)=-2.*P*(7.*U(122)-8.*U(121)+U(12Ø))/(X(61)-X(59))**2+
     1FI(U(61)-U(122))
      DO 2Ø I=2,6Ø,2
      Y(I)=-4.*EPS*P*(2.*U(I)-U(I-1)-U(I+1))/(X(I+1)-X(I-1))**2-
     1FI(U(I)-U(I+61))
      Y(I+61)=-4.*P*(2.*U(I+61)-U(I+6Ø)-U(I+62))/(X(I+1)-X(I-1))
     1**2+FI(U(I)-U(I+61))
   2Ø CONTINUE
      DO 3Ø I=3,59,2
      Y(I)=-2.*EPS*P*((7.*U(I)-8.*U(I-1)+U(I-2))/((X(I+2)-X(I-2))*
     1(X(I)-X(I-2)))+(7.*U(I)-8.*U(I+1)+U(I+2))/((X(I+2)-X(I-2))*
     2(X(I+2)-X(I))))-FI(U(I)-U(I+61))
      Y(I+61)=-2.*P*((7.*U(I+61)-8.*U(I+6Ø)+U(I+59))/((X(I+2)-X(I-2))*
     1(X(I)-X(I-2)))+(7.*U(I+61)-8.*U(I+62)+U(I+63))/((X(I+2)-X(I-2))*
     2(X(I+2)-X(I))))+FI(U(I)-U(I+61))
   3Ø CONTINUE
      RETURN
      END


C     PROGRAM TEST2(OUTPUT,TAPE6=OUTPUT)
C
C TEST PROGRAM 2 FOR J.G.VERWER'S ALGORITHM M3RK, AN EXPLICIT TIME-
C INTEGRATOR FOR SEMI-DISCRETE PARABOLIC EQUATIONS.
C THE TEST PROBLEM IS THE SINGLE TWO-DIMENSIONAL PARABOLIC EQUATION
C WHICH IS DISCUSSED IN SECTION 5.2 OF J.G.VERWER'S COMPANION PAPER
C "AN IMPLEMENTATION OF A CLASS OF STABILIZED, EXPLICIT METHODS FOR
C THE TIME INTEGRATION OF PARABOLIC EQUATIONS".
C THE TEST CONSISTS OF THE INTEGRATION OF THE SYSTEM OF 2Ø5 ODE'S
C WHICH IS CALLED SYSTEM II IN SECTION 5.2 OF THE COMPANION PAPER
C (EQUATIONS (5.6) WITH 5*41 GRID POINTS FOR THE GALERKIN DISCRETI-
C ZATION). IN THE PRESENT TEST THE TOLERANCE PARAMETER TOL=1.ØE-4.
C
C ***********************************************************************
C
C ARRAY DECLARATIONS FOR M3RK. THE ARRAY U IS THE INPUT ARRAY FOR THE
C INITIAL VECTOR AND UOUT IS THE OUTPUT ARRAY FOR THE COMPUTED SOLUT-
C ION VECTOR.
C DER IS THE SUBROUTINE DEFINING THE SYSTEM OF ODE'S.
C
      DIMENSION INFO(15),U(2Ø5),U1(2Ø5),U2(2Ø5),DU(2Ø5),DU1(2Ø5),
     +UOUT(2Ø5),SIGMA(2)
      EXTERNAL DER
C
C THE ARRAYS R AND Z WILL CONTAIN THE GRID POINTS USED BY THE GALERKIN
C DISCRETIZATION IN THE R-DIRECTION AND Z-DIRECTION,RESPECTIVELY.
C
      DIMENSION R(5),Z(41)
C
C ARRAY DECLARATIONS FOR THE AUXILARY SUBROUTINE EVAL WHICH GENERATES
C THE GALERKIN COEFFICIENTS OCCURRING IN THE SYTEM OF ODE'S(SEE EQUA-
C TIONS (5.6),(5.7) IN THE COMPANION PAPER). EVAL IS CALLED ONCE IN
C THE TEST PROGRAM.
C
      COMMON/ARRAY/DIAG(2Ø5),CO1(2Ø5),CO2(2Ø5),VRWZ(2Ø5),WZ(2Ø5)
C
C DECLARATION OF SOME CONSTANTS NEEDED IN DER. M IS THE NUMBER OF
C GRID POINTS IN THE R-DIRECTION, N IS THE NUMBER OF GRID POINTS
C IN THE Z-DIRECTION. REND IS THE RIGHT BOUNDARY OF THE R-INTERVAL
C
      COMMON/CONST/M,N,REND
C
C DEFINITION OF THE GRID FOR THE GALERKIN DISCRETIZATION. WE PUT
C M=5 AND N=41. THESE NUMBERS MAY BE CHANGED, PROVIDED THE LENGTH
C OF THE ARRAYS ARE ADJUSTED. THE VALUE OF N MINUS 1 MUST BE A
C MULTIPLE OF 4. THE INTEGER MN DEFINES THE NUMBER OF COMPONENTS
C OF THE SYSTEM OF ODE'S.
C
```

```
      REND=1.E-4
      ZEND=Ø.15
      N=41
      M=5
      MN=M*N
      DR = REND/FLOAT(M-1)
      DO 5 I = 1,M
    5 R(I) = DR*FLOAT(I-1)
      Z1=ZEND/1Ø.
      N1=(N-1)/4
      DZR=Z1/FLOAT(N1)
      DZM=4.*DZR
      N2=N1+1
      N3=N2+1
      N4=3*N2-2
      N5=N4+1
      DO 6 I=1,N2
    6 Z(I)=DZR*FLOAT(I-1)
      DO 7 I=N3,N4
    7 Z(I)=Z(I-1)+DZM
      DO 8 I=N5,N
    8 Z(I)=Z(I-1)+DZR
C
C PRINTING OF THE GRIDPOINTS.
C
      WRITE(6,2Ø)
   2Ø FORMAT(1H1,22H GRID POINTS ON R-AXIS)
      DO 22 I=1,M
      WRITE(6,21)R(I)
   21 FORMAT(1H ,E14.8)
   22 CONTINUE
      WRITE(6,23)
   23 FORMAT(1H1,22H GRID POINTS ON Z-AXIS)
      DO 25 I=1,N
      WRITE(6,24)Z(I)
   24 FORMAT(1H ,E14.8)
   25 CONTINUE
      WRITE(6,26)
   26 FORMAT(1H1)
C
C DEFINITION AND PRINTING OF THE INITIAL VECTOR OF THE SYSTEM OF
C ODE'S-UPRINT IS AN AUXILIARY PRINT ROUTINE-AND CALL OF EVAL.
C
      DO 3Ø I=1,M
      DO 3Ø J=1,N
      IJM=I+(J-1)*M
      U(IJM)=5ØØ.
   3Ø CONTINUE
      CALL UPRINT(M,N,MN,U,Z)
      CALL EVAL(R,M,Z,N,MN)
C
C DEFINITION OF PARAMETERS FOR M3RK. T IS THE INDEPENDENT VARIABLE.
C THE MEANING OF THE OTHER PARAMETERS IS CLEARLY EXPLAINED IN THE
C PROLOGUE OF COMMENTS OF M3RK.
C
      T=Ø.
      TOL=1.E-4
      INFO(1)=Ø
      INFO(2)=2
      INFO(3)=15ØØØ
C
C INITIAL AND SUBSEQUENT CALLS OF M3RK (SEE SECTION 5.2 OF THE COM-
C PANION PAPER). TE DEFINES THE POINT WHERE OUTPUT IS DESIRED.
C
      DO 9Ø I=1,7
      TE=Ø.1
      IF(I.EQ.2)TE=.2
      IF(I.EQ.3)TE=.4
      IF(I.EQ.4)TE=.6
      IF(I.EQ.5)TE=.8
      IF(I.EQ.6)TE=.9
      IF(I.EQ.7)TE=1.Ø
      CALL M3RK(T,TE,MN,H,HMIN,SIGMA,TOL,
     1DER,U,U1,U2,UOUT,DU,DU1,IFLAG,INFO)
C
C PRINTING OF THE OUTPUT POINT, OF THE ERROR FLAG, OF THE ESTIMATED
```

```
C SPECTRAL RADIUS, OF THE ARRAY INFO, AND OF THE COMPUTED SOLUTION.
C
      WRITE(6,5Ø)TE,IFLAG,SIGMA(1)
   5Ø FORMAT(1H1,13H ENDPOINT   =,E1Ø.4/11H  IFLAG =  ,I2/
     18H  SIGMA=,E13.7)
      WRITE(6,6Ø)(INFO(KK),KK=1,15)
   6Ø FORMAT(1H ,7H INFO= ,15(I5,1H )/)
      CALL UPRINT(M,N,MN,UOUT,Z)
   9Ø CONTINUE
      STOP
      END


      SUBROUTINE EVAL(R,M,Z,N,MN)
C
C THIS SUBROUTINE COMPUTES, IN A STRAIGHT FORWARD MANNER, THE STANDARD
C INTEGRALS GIVEN IN FORMULA (5.7) OF THE COMPANION PAPER. THE ARRAYS
C IN WHICH THE RESULTING VALUES ARE STORED, ARE USED IN DER. BY INSPEC-
C TION OF DER AND FORMULAS (5.6)-(5.7), THE DEFINITION OF THE ARRAYS
C WILL BECOME CLEAR.
C
      DIMENSION R(M),Z(N)
      COMMON/ARRAY/DIAG(2Ø5),CO1(2Ø5),CO2(2Ø5),VRWZ(2Ø5),WZ(2Ø5)
      WZ(1)=.Ø
      DO 1Ø I=1,MN
      DIAG(I)=.Ø
      CO1(I)=.Ø
      CO2(I)=.Ø
      VRWZ(I)=.Ø
   1Ø CONTINUE
      M1=M-1
      N1=N-1
      DO 4Ø I=1,M1
      RI=R(I)
      RIPLUS=R(I+1)
      DR=RIPLUS-RI
      WLEFT=DR*(2.*RI+RIPLUS)/6.
      WRGHT=DR*(RI+2.*RIPLUS)/6.
      WZ(I)=WZ(I)+WLEFT
      WZ(I+1)=WRGHT
   15 DO 3Ø J=1,N1
      ZJ=Z(J)
      ZJPLUS=Z(J+1)
      DZ=ZJPLUS-ZJ
      ALFA1=WLEFT/DZ
      ALFA2=WRGHT/DZ
      BETA=(WLEFT+WRGHT)*DZ/(2.Ø*DR*DR)
      VLEFT=WLEFT*DZ/2.
      VRGHT=WRGHT*DZ/2.
      L1=M*(J-1)+I
      L2=L1+1
      L3=L1+M
      L4=L3+1
      DIAG(L1)=DIAG(L1)+ALFA1+BETA
      DIAG(L2)=DIAG(L2)+ALFA2+BETA
      DIAG(L3)=DIAG(L3)+ALFA1+BETA
      DIAG(L4)=DIAG(L4)+ALFA2+BETA
      VRWZ(L1)=VRWZ(L1)+VLEFT
      VRWZ(L3)=VRWZ(L3)+VLEFT
      VRWZ(L2)=VRWZ(L2)+VRGHT
      VRWZ(L4)=VRWZ(L4)+VRGHT
      CO1(L1)=CO1(L1)-BETA
      CO2(L1)=CO2(L1)-ALFA1
      CO2(L2)=CO2(L2)-ALFA2
      CO1(L3)=CO1(L3)-BETA
   3Ø CONTINUE
   4Ø CONTINUE
      WZ1=WZ(1)
      DO 5Ø J=1,N
      JM1=(J-1)*M+1
   5Ø WZ(J)=VRWZ(JM1)/WZ1
      RETURN
      END
```

```
      SUBROUTINE DER(MN,P)
C
C THE PRESENT SUBROUTINE DEFINES THE SYSTEM OF ODE'S WE ARE INTEGRATING.
C THE ARRAYS DIAG,CO1,CO2,VRWZ AND WZ ARE DEFINED IN EVAL. THE ARRAYS
C P AND V ARE WORK ARRAYS.
C
      DIMENSION P(MN),V(2Ø5)
      COMMON/ARRAY/DIAG(2Ø5),CO1(2Ø5),CO2(2Ø5),VRWZ(2Ø5),WZ(2Ø5)
      COMMON/CONST/M,N,REND
C
      RLAM(X)=418.4*((((.1287496E-13*X-.116873E-9)*X+.384891E-6)
     1*X-.569812E-3)*X+.5718530 3)
      RHOC(X)=(146.44+.Ø18513*(X-1Ø4Ø.))*193ØØ.
      BRON(X)=7.1486**2*1.E-6*(-.3788Ø8E-1+.267688E-3*X+
     1.1724588E-7*X*X)/(3.141593**2*1.E-16)
      EPST(X)=((((-.27Ø491E-17*X+.3438691E-13)*X-.1639Ø5E-9)*
     1X+.33Ø5647E-6)*X-.1194E-3)*X+.Ø2667112
C
      DO 1Ø I=1,MN
   1Ø V(I)=P(I)
      DO 2Ø I=1,M
      P(I)=.Ø
      MNI=MN-M+I
      P(MNI)=.Ø
   2Ø CONTINUE
      M1=M+1
      MN1=MN-M
      DO 3Ø I=M1,MN1
      IMM=I-M
      IPM=I+M
      P(I)=-DIAG(I)*V(I)-CO1(I-1)*V(I-1)-CO1(I)*V(I+1)
     1-CO2(IMM)*V(IMM)-CO2(I)*V(IPM)
   3Ø CONTINUE
      N1=N-1
      DO 4Ø J=2,N1
      JTM=J*M
   4Ø P(JTM)=P(JTM)-REND*WZ(J)*EPST(V(JTM))/RLAM(V(JTM))
     1*5.6696E-8*V(JTM)**4
      DO 5Ø I=M1,MN1
   5Ø P(I)=(P(I)*RLAM(V(I))/VRWZ(I)+BRON(V(I)))/RHOC(V(I))
      RETURN
      END


      SUBROUTINE UPRINT(M,N,MN,U,Z)
C
C A PRINT ROUTINE FOR THE COMPUTED SOLTUION VECTOR.
C
      DIMENSION U(MN),Z(N)
      WRITE(6,1)
    1 FORMAT(///4X,1HZ,4ØX,8HU(T,R,Z)/)
      DO 1Ø I=1,N
      K=(I-1)*M
      K1=K+1
      KM=K+M
   1Ø WRITE(6,2Ø)Z(I),(U(J),J=K1,KM)
   2Ø FORMAT(F1Ø.6,5X,5F13.6)
      RETURN
      END


      SUBROUTINE M3RK(X,XE,N,H,HMIN,SIGMA,TOL,F,Y,                    1Ø
     +                Y1,Y2,YXE,DY,DY1,IFLAG,INFO)                    2Ø
C*********************************************************************  3Ø
C* ABSTRACT                                                         *  4Ø
C*********************************************************************  5Ø
C* M3RK IS DESIGNED TO SOLVE INITIAL VALUE PROBLEMS FOR SYSTEMS OF  *  6Ø
C* ORDINARY DIFFERENTIAL EQUATIONS OF THE FORM                      *  7Ø
C*                   DY/DX=F(Y(1),...,Y(N)),                        *  8Ø
C*                   Y(I) GIVEN AT X,                               *  9Ø
C* WHICH ORIGINATE FROM SEMI-DISCRETIZATION OF INITIAL-BOUNDARY VALUE *  1ØØ
C* PROBLEMS FOR PARABOLIC PARTIAL DIFFERENTIAL EQUATIONS. M3RK IS   *  11Ø
C* BASED ON STABILIZED,EXPLICIT THREE-STEP RUNGE-KUTTA FORMULAS OF  *  12Ø
C* ORDER ONE AND TWO,OF WHICH THE DEGREE CAN VARY BETWEEN 2 AND 12. *  13Ø
C*                                                                  *  14Ø
```

```
C* M3RK NEEDS 6 ARRAYS OF LENGTH N,WHICH ALL APPEAR IN THE CALL LIST.  *   15Ø
C* THE CODE INTEGRATES FROM X TO XE.ON NORMAL RETURN THE PARAMETERS IN *   16Ø
C* THE CALL LIST ARE READY FOR CONTINUING THE INTEGRATION.TO CONTINUE  *   17Ø
C* THE INTEGRATION,THE USER NEEDS ONLY TO REDEFINE THE OUTPUT POINT XE *   18Ø
C* AND CALL AGAIN.                                                     *   19Ø
C*                                                                     *   2ØØ
C* M3RK CALLS 11 SUBROUTINES WHICH HAVE BEEN WRITTEN TO STRUCTURE THE  *   21Ø
C* PROGRAM.THESE SUBROUTINES ARE:                                      *   22Ø
C* HSTART - HSTART COMPUTES THE INITIAL STEPLENGTH                     *   23Ø
C* PARAM  - PARAM COMPUTES PARAMETERS OF THE VARIOUS IMPLEMENTED       *   24Ø
C*          SCHEMES FROM THE COEFFICIENTS OF STABILITY POLYNOMIALS     *   25Ø
C* POWERM - POWERM ESTIMATES THE SPECTRAL RADIUS OF THE JACOBIAN OF F  *   26Ø
C* MAXDEG - MAXDEG COMPUTES THE MAXIMAL DEGREE OF THE FORMULAS,        *   27Ø
C*          WHICH IS ALLOWED WITH RESPECT TO INTERNAL STABILITY        *   28Ø
C* MINDEG - MINDEG COMPUTES THE MINIMAL DEGREE OF THE FORMULAS,        *   29Ø
C*          WHICH IS ALLOWED WITH RESPECT TO ABSOLUTE STABILITY        *   3ØØ
C* STEP   - STEP CONTAINS THE ACTUAL INTEGRATOR                        *   31Ø
C* ESTIMA - ESTIMA COMPUTES A LOCAL ERROR BOUND AND ESTIMATES A LOCAL  *   32Ø
C*          ERROR                                                      *   33Ø
C* NEWH   - NEWH DELIVERS A NEW STEPLENGTH                             *   34Ø
C* INTER1 - INTER1 PERFORMS INTERPOLATION AFTER A CHANGE OF THE        *   35Ø
C*          STEPLENGTH                                                 *   36Ø
C* INTER2 - INTER2 INTERPOLATES THE SOLUTION AT THE OUTPUT POINT XE    *   37Ø
C* SHIFT  - SHIFT SHIFTS THE DATA FOR A NEXT STEP                      *   38Ø
C*                                                                     *   39Ø
C* THESE SUBROUTINES ARE COMPLETELY LOCAL,I.E.THE INFORMATION THEY     *   4ØØ
C* NEED IS PASSED THROUGH THE PARAMETER LISTS.THE WHOLE PACKAGE HAS    *   41Ø
C* BEEN TESTED ON A CDC CYBER 73-28 USING AN ARITHMETIC PRECISION OF   *   42Ø
C* 14 DIGITS.THE CODE POWERM USES THE CDC SYSTEM SUBPROGRAMS RANSET    *   43Ø
C* AND RANF CONSTITUTING A RANDOM GENERATOR.RANSET AND RANF MUST BE    *   44Ø
C* REPLACED WHEN USING THE PROGRAM ON ANOTHER COMPUTER.                *   45Ø
C* THE CODES CALLED BY M3RK USE NO MACHINE DEPENDENT CONSTANTS.        *   46Ø
C* M3RK USES ONE MACHINE DEPENDENT CONSTANT,NAMELY THE ARITHMETIC      *   47Ø
C* PRECISION OF THE COMPUTER.IN THE PROGRAM THE INTERNAL VARIABLE APR, *   48Ø
C* WHICH REPRESENTS THE ARITHMETIC PRECISION,EQUALS 1.ØE-14. APR MUST  *   49Ø
C* BE CHANGED ACCORDINGLY WHEN USING THE PROGRAM ON ANOTHER COMPUTER.  *   5ØØ
C*                                                                     *   51Ø
C* THE WHOLE PROGRAM PACKAGE IS ACCEPTED BY THE PFORT VERIFIER.THE     *   52Ø
C* PFORT VERIFIER IS A PROGRAM WHICH CHECKS A FORTRAN PROGRAM,I.E. A   *   53Ø
C* MAIN PROGRAM AND SUBPROGRAMS,FOR ADHERENCE TO PFORT,A PORTABLE      *   54Ø
C* SUBSET OF AMERICAN NATIONAL STANDARD FORTRAN(SEEI1 ).THE WHOLE      *   55Ø
C* PROGRAM PACKAGE IS COMPLETELY EXPLAINED AND DESCRIBED IN I2 .       *   56Ø
C*********************************************************************   57Ø
C* MEANING OF THE PARAMETERS                                           *   58Ø
C*********************************************************************   59Ø
C* X     - VARIABLE    : INDEPENDENT VARIABLE                          *   6ØØ
C* XE    - EXPRESSION  : OUTPUT POINT AT WHICH SOLUTION IS DESIRED     *   61Ø
C* N     - EXPRESSION  : NUMBER OF EQUATIONS                           *   62Ø
C* H     - VARIABLE    : STEPLENGTH                                    *   63Ø
C* HMIN  - VARIABLE    : MINIMAL STEPLENGTH                            *   64Ø
C* SIGMA - ARRAY       : AN ARRAY OF LENGTH 2 CONTAINING ESTIMATES     *   65Ø
C*                       OF THE SPECTRAL RADIUS OF THE JACOBIAN OF F   *   66Ø
C* TOL   - EXPRESSION  : LOCAL ERROR TOLERANCE                         *   67Ø
C* F     - SUBROUTINE  : DERIVATIVE                                    *   68Ø
C* Y     - ARRAY       : SOLUTION VECTOR AT X,INPUT AND WORK ARRAY     *   69Ø
C* Y1    - ARRAY       : SOLUTION VECTOR AT X-H,WORK ARRAY             *   7ØØ
C* Y2    - ARRAY       : SOLUTION VECTOR AT X-2H,WORK ARRAY            *   71Ø
C* YXE   - ARRAY       : SOLUTION VECTOR AT XE,OUTPUT AND WORK ARRAY   *   72Ø
C* DY    - ARRAY       : DERIVATIVE VECTOR AT X,WORK ARRAY             *   73Ø
C* DY1   - ARRAY       : DERIVATIVE VECTOR AT X-H,WORK ARRAY           *   74Ø
C* IFLAG - VARIABLE    : ERROR FLAG                                    *   75Ø
C* INFO  - ARRAY       : INTEGER ARRAY OF LENGTH 15.INFO IS USED TO    *   76Ø
C*                       PASS INFORMATION TO INITIALIZE THE CODE,TO    *   77Ø
C*                       PASS INFORMATION BETWEEN THE MAIN PROGRAM AND *   78Ø
C*                       THE SUBPROGRAMS,TO PASS INFORMATION TO THE    *   79Ø
C*                       USER ABOUT THE STATUS OF THE INTEGRATION,AND  *   8ØØ
C*                       TO RETAIN INFORMATION FOR SUBSEQUENT CALLS.   *   81Ø
C*********************************************************************   82Ø
C* FIRST CALL TO M3RK                                                  *   83Ø
C*********************************************************************   84Ø
C* THE USER MUST PROVIDE STORAGE IN HIS CALLING PROGRAM FOR THE ARRAYS *   85Ø
C* IN THE CALL LIST.HE HAS TO SUPPLY THE SUBROUTINE F(N,Y) FOR         *   86Ø
C* EVALUATING THE DERIVATIVES DY(I)/DT,I=1,...,N,WHICH MUST BE OVER-   *   87Ø
C* WRITTEN ON Y(I).FOR THE SPECTRAL RADIUS THERE EXIST THREE OPTIONS   *   88Ø
C* WHICH MUST BE SELECTED WITH INFO(2).IN INFO(3) THE USER MUST GIVE   *   89Ø
```

```
C* A MAXIMUM FOR THE NUMBER OF EVALUATIONS OF F(Y) TO BE SPENT(ON RE-  *   9ØØ
C* TURN IT MAY OCCUR THAT INFO(3) IS EXCEEDED WITH ABOUT 5Ø)           *   91Ø
C************************                                             *   92Ø
C* INPUT PARAMETERS                                                   *   93Ø
C************************                                             *   94Ø
C* X     - INITIAL VALUE OF THE INDEPENDENT VARIABLE                  *   95Ø
C* XE    - OUTPUT POINT AT WHICH SOLUTION IS DESIRED                  *   96Ø
C* N     - NUMBER OF EQUATIONS                                        *   97Ø
C* SIGMA - (1)= AN UPPER ESTIMATION OF THE SPECTRAL RADIUS OF THE JA- *   98Ø
C*             COBIAN OF F IN CASE OF OPTION 1.FOR OPTION 2 AND 3 NO *   99Ø
C*             INITIALIZATION IS REQUIRED                            *  1ØØØ
C* TOL   - LOCAL ERROR TOLERANCE                                      *  1Ø1Ø
C* Y     - VECTOR OF INITIAL VALUES OF THE DEPENDENT VARIABLES        *  1Ø2Ø
C* INFO  - (1)= Ø TO INDICATE FIRST CALL                              *  1Ø3Ø
C*         (2)= 1 TO INDICATE OPTION 1 FOR THE SPECTRAL RADIUS,I.E.   *  1Ø4Ø
C*              THE USER MUST INITIALIZE SIGMA(1)                     *  1Ø5Ø
C*            = 2 TO INDICATE OPTION 2 FOR THE SPECTRAL RADIUS,I.E.   *  1Ø6Ø
C*              THE CODE INITIALIZES SIGMA(1) AND CONTROLS SIGMA(1)   *  1Ø7Ø
C*            = 3 TO INDICATE OPTION 3 FOR THE SPECTRAL RADIUS,I.E.   *  1Ø8Ø
C*              THE CODE ONLY INITIALIZES SIGMA(1) AT THE FIRST CALL  *  1Ø9Ø
C*         (3)= MAXIMUM NUMBER OF F(Y) EVALUATIONS TO BE SPENT        *  11ØØ
C************************                                             *  111Ø
C* OUTPUT PARAMETERS                                                  *  112Ø
C************************                                             *  113Ø
C* X     - LAST POINT REACHED IN INTEGRATION.NORMALLY X IS SLIGHTLY   *  114Ø
C*         BEYOND XE                                                  *  115Ø
C* H     - INITIAL STEPLENGTH FOR SUBSEQUENT CALL                     *  116Ø
C* HMIN  - MINIMAL STEPLENGTH USED BY M3RK                            *  117Ø
C* SIGMA - (1)= UPPER ESTIMATION OF THE SPECTRAL RADIUS INITIALIZED BY *  118Ø
C*             THE USER OR BY POWERM                                  *  119Ø
C*       - (2)= INACCURATE ESTIMATION OF THE SPECTRAL RADIUS USED FOR  *  12ØØ
C*             ITS CONTROL                                            *  121Ø
C* Y     - SOLUTION VECTOR AT X                                       *  122Ø
C* Y1    - SOLUTION VECTOR AT X-H                                     *  123Ø
C* Y2    - SOLUTION VECTOR AT X-2H                                    *  124Ø
C* YXE   - SOLUTION VECTOR AT OUTPUT POINT XE                         *  125Ø
C* DY    - DERIVATIVE VECTOR AT X                                     *  126Ø
C* DY1   - DERIVATIVE VECTOR AT X-H                                   *  127Ø
C* IFLAG - = Ø NORMAL RETURN,I.E.OUTPUT POINT IS REACHED              *  128Ø
C*         = 1 OUTPUT POINT IS NOT REACHED.THE MAXIMUM NUMBER OF      *  129Ø
C*             F(Y)-EVALUATIONS HAS BEEN SPENT.THE PROCESS CAN BE CON- *  13ØØ
C*             TINUED BY INCREASING INFO(3) AND CALLING AGAIN         *  131Ø
C*         = 2 MAXIMAL DEGREE FALLS OUTSIDE THE RANGE AS TOL/APR IS   *  132Ø
C*             TOO SMALL.THE PROCESS IS NOT STARTED                   *  133Ø
C*         = 3 POWERM FAILED IN THE ESTIMATION OF THE SPECTRAL RADIUS. *  134Ø
C*             THE PROCESS IS DISCONTINUED                            *  135Ø
C* INFO  - (1) = 1 TO INDICATE THAT THE NEXT CALL IS A SUBSEQUENT ONE *  136Ø
C*         (2) = 1 IN CASE OF OPTION 1 OR 3,ELSE 2                    *  137Ø
C*         (3) = UNCHANGED                                            *  138Ø
C*         (4) = TOTAL NUMBER OF INTEGRATION STEPS PERFORMED,I.E.     *  139Ø
C*               ACCEPTED AND REJECTED ONES                           *  14ØØ
C*         (5) = NUMBER OF REJECTED INTEGRATION STEPS                 *  141Ø
C*         (6) = NUMBER OF RESTARTS INITIATED BY THE CODE             *  142Ø
C*         (7) = TOTAL NUMBER OF DERIVATIVE EVALUATIONS               *  143Ø
C*         (8) = NUMBER OF DERIVATIVE EVALUATIONS USED FOR THE ESTIMA- *  144Ø
C*               TION AND CONTROL OF THE SPECTRAL RADIUS              *  145Ø
C*         (9) = CURRENT DEGREE                                       *  146Ø
C*         (1Ø)= MAXIMAL DEGREE FOR THE FIRST ORDER FORMULAS          *  147Ø
C*         (11)= MAXIMAL DEGREE FOR THE SECOND ORDER FORMULAS         *  148Ø
C*         (12)= CURRENT ORDER                                        *  149Ø
C*         (13)= NUMBER OF STEPS PERFORMED AFTER START OR RESTART     *  15ØØ
C*         (14)= NUMBER OF STEPS PERFORMED AFTER CHANGE OF H OR ORDER *  151Ø
C*         (15)= NUMBER OF STEPS PERFORMED AFTER ESTIMATION OF SPEC-  *  152Ø
C*               TRAL RADIUS                                          *  153Ø
C*                                                                    *  154Ø
C* IT IS EMPHASIZED THAT THE OUTPUT LIST GIVEN ABOVE IS NOT STRICTLY  *  155Ø
C* VALID WHEN ON RETURN IFLAG IS EQUAL TO 2 OR 3.                     *  156Ø
C* IT IS FURTHER EMPHASIZED THAT ON RETURN THE ARRAY Y ALWAYS CONTAINS *  157Ø
C* THE SOLUTION VECTOR AT THE POINT X.THUS WHEN IFLAG=3,THE USER HAS  *  158Ø
C* THE POSSIBILITY TO RESTART THE PROCESS AT THE POINT X,PROVIDED HE  *  159Ø
C* IS ABLE TO TAKE ACTION WITH RESPECT TO THE ESTIMATION OF THE SPEC- *  16ØØ
C* TRAL RADIUS.                                                       *  161Ø
C*********************************************************************  162Ø
C* SUBSEQUENT CALLS TO M3RK                                           *  163Ø
```

```
C*************************************************************************  164Ø
C* ON RETURN OF M3RK ALL PARAMETERS ARE READY FOR CONTINUING THE INTE- *  165Ø
C* GRATION,PROVIDED IFLAG IS NOT EQUAL TO 2 OR 3.IF XE IS REACHED AND  *  166Ø
C* A NORMAL CONTINUATION IS DESIRED,THE USER NEEDS ONLY TO DEFINE A NEW*  167Ø
C* OUTPUT POINT XE AND CALL AGAIN.IF ON RETURN IFLAG=1 AND THE USER    *  168Ø
C* WANTS TO CONTINUE,HE ONLY NEEDS TO INCREASE INFO(3) AND CALL AGAIN. *  169Ø
C* THE PROGRAM IS WRITTEN IN SUCH A WAY THAT THE CHOICE OF OUTPUT      *  17ØØ
C* POINTS DOES NOT AFFECT THE INTEGRATION PROCESS ITSELF.BETWEEN SUB-  *  171Ø
C* SEQUENT CALLS THE USER MAY INCREASE TOL.ALL OTHER PARAMETERS MUST   *  172Ø
C* REMAIN UNCHANGED.THE COUNTERS IN INFO ARE USED ACCUMULATIVELY.      *  173Ø
C*************************************************************************  174Ø
C*  PROGRAM TEXT                                                       *  175Ø
C*************************************************************************  176Ø
      DIMENSION Y(N),Y1(N),Y2(N),YXE(N),DY(N),DY1(N),SIGMA(2),INFO(15)     177Ø
C**********************************                                      *  178Ø
C*  MEANING OF THE INTERNAL VARIABLES                                  *  179Ø
C**********************************                                      *  18ØØ
C*  ALFA   - THE FACTOR FOR CHANGING THE STEPLENGTH                    *  181Ø
C*  APR    - THE ARITHMETIC PRECISION                                  *  182Ø
C*  HOLD   - PREVIOUSLY ACCEPTED STEPLENGTH                           *  183Ø
C*  MOLD   - PREVIOUSLY USED DEGREE                                   *  184Ø
C*  REJECT - NUMBER OF SUCCESSIVE STEP FAILURES                        *  185Ø
C*  B      - NONZERO B-PARAMETERS OF THE SCHEME                        *  186Ø
C*  C      - C-PARAMETERS OF THE SCHEME                                *  187Ø
C*  LA     - LABDA-PARAMETERS OF THE SCHEME                            *  188Ø
C*  HMAX   - MAXIMAL STEPSIZES WITH RESPECT TO ABSOLUTE STABILITY      *  189Ø
C*  EPS    - LOCAL ERROR BOUND                                         *  19ØØ
C*  ERROR  - ESTIMATED LOCAL ERROR                                     *  191Ø
C*************************************************************************  192Ø
      INTEGER REJECT                                                       193Ø
      REAL LA                                                              194Ø
      DIMENSION B(2),C(12),LA(12),HMAX(2)                                  195Ø
      EXTERNAL F                                                           196Ø
C*************************************************************************  197Ø
C*  IF ALREADY PAST OUTPUT POINT DURING A SUBSEQUENT CALL,THEN INTER-  *  198Ø
C*  POLATE AND RETURN                                                  *  199Ø
C*************************************************************************  2ØØØ
      IF(INFO(1).EQ.Ø.OR.X.LT.XE) GOTO 1Ø                                  2Ø1Ø
      CALL INTER2(N,Y,Y1,Y2,YXE,(X-XE)/H)                                  2Ø2Ø
      RETURN                                                               2Ø3Ø
C*************************************************************************  2Ø4Ø
C*  SET THE ERROR FLAG IFLAG EQUAL TO ZERO AND INITIALIZE APR.DETERMINE *  2Ø5Ø
C*  THE MAXIMAL DEGREES WITH RESPECT TO INTERNAL STABILITY.IF NECESSARY *  2Ø6Ø
C*  INTERRUPT                                                          *  2Ø7Ø
C*************************************************************************  2Ø8Ø
   1Ø IFLAG=Ø                                                              2Ø9Ø
      APR=1.ØE-14                                                          21ØØ
      CALL MAXDEG(TOL,APR,IFLAG,INFO)                                      211Ø
      IF(IFLAG.NE.2) GOTO 2Ø                                               212Ø
      RETURN                                                               213Ø
C*************************************************************************  214Ø
C*  SET THE CONTROL VARIABLES INFO(9),REJECT AND HOLD FOR A CONTINUING  *  215Ø
C*  CALL,AND INITIALIZE THE ARRAY HMAX                                 *  216Ø
C*************************************************************************  217Ø
   2Ø INFO(9)=Ø                                                            218Ø
      IF(INFO(1).EQ.Ø) GOTO 3Ø                                             219Ø
      REJECT=Ø                                                             22ØØ
      HOLD=H                                                               221Ø
      HMAX(1)=5.15*FLOAT(INFO(1Ø))*FLOAT(INFO(1Ø))/SIGMA(1)                222Ø
      HMAX(2)=2.29*FLOAT(INFO(11))*FLOAT(INFO(11))/SIGMA(1)                223Ø
      GOTO 8Ø                                                              224Ø
C*************************************************************************  225Ø
C*  SET REJECT,HOLD AND H FOR THE FIRST CALL. SET THE ELEMENTS INFO(I), *  226Ø
C*  I=4,...,8,13,14,15 EQUAL TO ZERO AND INFO(12) EQUAL TO 2.TO PREPARE *  227Ø
C*  THE FIRST STEP EVALUATE F(Y) AND SUBSTITUTE INTO DY.IF NECESSARY    *  228Ø
C*  ESTIMATE THE SPECTRAL RADIUS AND CHECK FOR A FAILURE OF THE POWER   *  229Ø
C*  METHOD.INITIALIZE ARRAY HMAX AND ESTIMATE THE INITIAL STEPLENGTH    *  23ØØ
C*************************************************************************  231Ø
   3Ø INFO(1)=1                                                            232Ø
      REJECT=Ø                                                             233Ø
      DO 4Ø I=4,8                                                          234Ø
   4Ø INFO(I)=Ø                                                            235Ø
      DO 5Ø I=13,15                                                        236Ø
   5Ø INFO(I)=Ø                                                            237Ø
      INFO(12)=2                                                           238Ø
      DO 6Ø I=1,N                                                          239Ø
```

```
      DY(I)=Y(I)                                                        24ØØ
      DY1(I)=Ø.                                                         241Ø
      Y1(I)=Ø.                                                          242Ø
      Y2(I)=Ø.                                                          243Ø
   6Ø CONTINUE                                                          244Ø
      CALL F(N,DY)                                                      245Ø
      INFO(7)=INFO(7)+1                                                 246Ø
      IF(INFO(2).EQ.1) GOTO 7Ø                                          247Ø
      SIGMA(1)=Ø.                                                       248Ø
      CALL POWERM(N,Y,Y1,YXE,DY,DY1,F,SIGMA,APR,IFLAG,INFO)             249Ø
      IF(IFLAG.NE.3) GOTO 7Ø                                            25ØØ
      RETURN                                                            251Ø
C                                                                       252Ø
   7Ø HMAX(1)=5.15*FLOAT(INFO(1Ø))*FLOAT(INFO(1Ø))/SIGMA(1)             253Ø
      HMAX(2)=2.29*FLOAT(INFO(11))*FLOAT(INFO(11))/SIGMA(1)             254Ø
      CALL HSTART(N,Y,DY,YXE,F,TOL,APR,SIGMA(1),HMIN,INFO)              255Ø
      H=HMIN                                                            256Ø
      HOLD=H                                                            257Ø
C*********************************************************************  258Ø
C* DETERMINE THE DEGREE,AND,IF NECESSARY,CALCULATE THE PARAMETERS OF   *  259Ø
C* THE SCHEME TO BE USED                                               *  26ØØ
C*********************************************************************  261Ø
   8Ø MOLD=INFO(9)                                                      262Ø
      CALL MINDEG(H,SIGMA(1),INFO)                                      263Ø
      IF(INFO(9).EQ.MOLD) GOTO 9Ø                                       264Ø
      CALL PARAM(C,LA,B,INFO)                                           265Ø
C*********************************************************************  266Ø
C* CHECK IF THE MAXIMUM NUMBER OF EVALUATIONS IS REACHED.UPDATE HMIN   *  267Ø
C* AND CALCULATE A SOLUTION AT X+H                                     *  268Ø
C*********************************************************************  269Ø
   9Ø IF(INFO(7).GE.INFO(3)) IFLAG=1                                    27ØØ
      IF(IFLAG.NE.1) GOTO 1ØØ                                           271Ø
      RETURN                                                            272Ø
C                                                                       273Ø
  1ØØ IF(H.LT.HMIN) HMIN=H                                              274Ø
      CALL STEP(N,Y,Y1,Y2,YXE,DY,DY1,H,F,C,LA,B,INFO)                   275Ø
      INFO(13)=INFO(13)+1                                               276Ø
      INFO(4)=INFO(4)+1                                                 277Ø
C*********************************************************************  278Ø
C* IF THE PROCESS IS IN THE START PHASE,SHIFT THE DATA.CHECK FOR THE   *  279Ø
C* CONTINUATION WITH A THREE-STEP SCHEME.IF THE OUTPUT POINT IS PASSED *  28ØØ
C* INTERPOLATE AND RETURN                                              *  281Ø
C*********************************************************************  282Ø
      IF(INFO(13).GE.3) GOTO 11Ø                                        283Ø
      CALL SHIFT(N,Y,Y1,Y2,YXE,DY,DY1,X,HOLD,F,INFO)                    284Ø
      INFO(14)=INFO(14)+1                                               285Ø
      INFO(15)=INFO(15)+1                                               286Ø
      IF(INFO(13).EQ.1) GOTO 9Ø                                         287Ø
      INFO(9)=Ø                                                         288Ø
      IF(X.LT.XE) GOTO 8Ø                                               289Ø
      CALL INTER2(N,Y,Y1,Y2,YXE,(X-XE)/HOLD)                            29ØØ
      RETURN                                                            291Ø
C*********************************************************************  292Ø
C* CALCULATE THE LOCAL ERRORBOUND AND ESTIMATE THE LOCAL ERROR.        *  293Ø
C*********************************************************************  294Ø
  11Ø CALL ESTIMA(N,Y,Y1,Y2,YXE,TOL,EPS,ERROR,INFO)                     295Ø
C*********************************************************************  296Ø
C* IF THE ERROR IS TOO LARGE FOR THE THIRD STEP AFTER START,THEN RE-   *  297Ø
C* START AT INITIAL POINT WITH H=H/1Ø                                  *  298Ø
C*********************************************************************  299Ø
      IF(EPS.GE.ERROR.OR.INFO(13).NE.3) GOTO 13Ø                        3ØØØ
      INFO(6)=INFO(6)+1                                                 3Ø1Ø
      INFO(5)=INFO(5)+3                                                 3Ø2Ø
      INFO(9)=Ø                                                         3Ø3Ø
      INFO(13)=Ø                                                        3Ø4Ø
      INFO(14)=Ø                                                        3Ø5Ø
      INFO(15)=Ø                                                        3Ø6Ø
      X=X-2.*H                                                          3Ø7Ø
      H=H/1Ø.                                                           3Ø8Ø
      HOLD=H                                                            3Ø9Ø
      DO 12Ø I=1,N                                                      31ØØ
      Y(I)=Y2(I)                                                        311Ø
      DY(I)=Y(I)                                                        312Ø
  12Ø CONTINUE                                                          313Ø
      CALL F(N,DY)                                                      314Ø
      INFO(7)=INFO(7)+1                                                 315Ø
```

```
      GOTO 8Ø                                                          316Ø
C*********************************************************************  317Ø
C* IF STEP FAILED,CHECK FOR A REESTIMATION OF THE SPECTRAL RADIUS.IF   *  318Ø
C* NECESSARY,CHECK FOR A FAILURE OF POWERM AND UPDATE HMAX             *  319Ø
C*********************************************************************  32ØØ
  13Ø IF(INFO(2).EQ.1.OR.INFO(15).EQ.Ø.OR.EPS.GE.ERROR) GOTO 15Ø        321Ø
      SIGMA(1)=Ø.                                                      322Ø
      CALL POWERM(N,Y,Y1,YXE,DY,DY1,F,SIGMA,APR,IFLAG,INFO)            323Ø
      IF(IFLAG.NE.3) GOTO 14Ø                                          324Ø
      RETURN                                                           325Ø
C                                                                      326Ø
  14Ø HMAX(1)=5.15*FLOAT(INFO(1Ø))*FLOAT(INFO(1Ø))/SIGMA(1)            327Ø
      HMAX(2)=2.29*FLOAT(INFO(11))*FLOAT(INFO(11))/SIGMA(1)            328Ø
C*********************************************************************  329Ø
C* CALCULATE A NEW STEPLENGTH                                         *  33ØØ
C*********************************************************************  331Ø
  15Ø HOLD=H                                                           332Ø
      CALL NEWH(EPS/ERROR,HOLD,H,ALFA,HMAX,INFO)                       333Ø
C*********************************************************************  334Ø
C* IF THE ORDER EQUALS 1 AND THE STEPLENGTH IS SMALLER THAN THE MAXI-  *  335Ø
C* MAL STEPLENGTH FOR ORDER 2 RESET THE ORDER                          *  336Ø
C*********************************************************************  337Ø
      IF(INFO(12).EQ.1.AND.H.LT.HMAX(2)) INFO(9)=Ø                     338Ø
      IF(INFO(9).EQ.Ø) INFO(12)=2                                      339Ø
C*********************************************************************  34ØØ
C* IF STEP FAILED REJECT THE INTEGRATION STEP.CHECK FOR THREE SUCCES-  *  341Ø
C* SIVE FAILURES,AND,IF NECESSARY,INTERPOLATE FOR THE NEW STEPLENGTH   *  342Ø
C* *******************************************************************  343Ø
      IF(EPS.GE.ERROR) GOTO 17Ø                                        344Ø
      REJECT=REJECT+1                                                  345Ø
      INFO(5)=INFO(5)+1                                                346Ø
      IF(REJECT.EQ.3) GOTO 16Ø                                         347Ø
      CALL INTER1(N,Y,Y1,Y2,DY1,F,ALFA,INFO)                           348Ø
      GOTO 8Ø                                                          349Ø
C*********************************************************************  35ØØ
C* RESET REJECT,INFO(I),I=6,9,12,13,14 AND THE STEPLENGTH FOR A        *  351Ø
C* RESTART                                                             *  352Ø
C*********************************************************************  353Ø
  16Ø REJECT=Ø                                                         354Ø
      INFO(6)=INFO(6)+1                                                355Ø
      INFO(9)=Ø                                                        356Ø
      INFO(12)=2                                                       357Ø
      INFO(13)=Ø                                                       358Ø
      INFO(14)=Ø                                                       359Ø
      CALL HSTART(N,Y,DY,YXE,F,TOL,APR,SIGMA(1),H,INFO)                36ØØ
      HOLD=H                                                           361Ø
      GOTO 8Ø                                                          362Ø
C*********************************************************************  363Ø
C* FIND OUT IF THE ORDER SHOULD CHANGE FROM 2 TO 1.ON EXIT OF THIS     *  364Ø
C* PROGRAM PART H SHOULD BE RESET TO HMAX(2).IF THE ORDER HAS TO BE    *  365Ø
C* CHANGED FROM 2 TO 1 SET INFO(9)=Ø                                   *  366Ø
C*********************************************************************  367Ø
  17Ø IF(INFO(14).LT.3.OR.INFO(12).EQ.1) GOTO 18Ø                      368Ø
      IF(HOLD.NE.HMAX(2).OR.H.NE.HMAX(2)) GOTO 18Ø                     369Ø
      INFO(12)=1                                                       37ØØ
      CALL ESTIMA(N,Y,Y1,Y2,YXE,TOL,EPS,ERROR,INFO)                    371Ø
      CALL NEWH(EPS/ERROR,HOLD,H,ALFA,HMAX,INFO)                       372Ø
      IF(ALFA.LE.1.) INFO(12)=2                                        373Ø
      H=HMAX(2)                                                        374Ø
      INFO(14)=-1                                                      375Ø
      IF(INFO(12).EQ.2) GOTO 18Ø                                       376Ø
      INFO(9)=Ø                                                        377Ø
C*********************************************************************  378Ø
C* SHIFT THE DATA                                                      *  379Ø
C*********************************************************************  38ØØ
  18Ø CALL SHIFT(N,Y,Y1,Y2,YXE,DY,DY1,X,HOLD,F,INFO)                   381Ø
      REJECT=Ø                                                         382Ø
      INFO(14)=INFO(14)+1                                              383Ø
      INFO(15)=INFO(15)+1                                              384Ø
C*********************************************************************  385Ø
C* CHECK FOR A REESTIMATION OF THE SPECTRAL RADIUS.IF NECESSARY,CHECK  *  386Ø
C* FOR A FAILURE OF POWERM AND UPDATE HMAX                             *  387Ø
C*********************************************************************  388Ø
      IF(INFO(2).EQ.1.OR.INFO(15).NE.25) GOTO 2ØØ                      389Ø
      CALL POWERM(N,Y,Y1,YXE,DY,DY1,F,SIGMA,APR,IFLAG,INFO)            39ØØ
      IF(IFLAG.NE.3) GOTO 19Ø                                          391Ø
```

```
      RETURN                                                            392Ø
C                                                                       393Ø
  19Ø HMAX(1)=5.15*FLOAT(INFO(1Ø))*FLOAT(INFO(1Ø))/SIGMA(1)             394Ø
      HMAX(2)=2.29*FLOAT(INFO(11))*FLOAT(INFO(11))/SIGMA(1)             395Ø
C*************************************************************************396Ø
C* IF X IS SMALLER THAN XE CONTINUE THE INTEGRATION.IF NECESSARY,IN-   * 397Ø
C* TERPOLATE FOR A CHANGE OF H                                         * 398Ø
C*************************************************************************399Ø
  2ØØ IF(X.GE.XE) GOTO 21Ø                                              4ØØØ
      IF(H.NE.HOLD) CALL INTER1(N,Y,Y1,Y2,DY1,F,ALFA,INFO)              4Ø1Ø
      IF(H.NE.HOLD.OR.INFO(9).EQ.Ø) GOTO 8Ø                             4Ø2Ø
      GOTO 9Ø                                                           4Ø3Ø
C*************************************************************************4Ø4Ø
C* INTERPOLATE AT XE AND,IF NECESSARY,INTERPOLATE FOR A CHANGE OF H    * 4Ø5Ø
C* BEFORE RETURN                                                       * 4Ø6Ø
C*************************************************************************4Ø7Ø
  21Ø CALL INTER2(N,Y,Y1,Y2,YXE,(X-XE)/HOLD)                            4Ø8Ø
      IF(H.NE.HOLD) CALL INTER1(N,Y,Y1,Y2,DY1,F,ALFA,INFO)              4Ø9Ø
      RETURN                                                            41ØØ
      END                                                               411Ø


      SUBROUTINE HSTART(N,Y,DY,YXE,F,TOL,APR,SIGMA,H,INFO)              412Ø
C*************************************************************************413Ø
C* SUBROUTINE HSTART CALCULATES THE INITIAL STEPLENGTH                 * 414Ø
C*************************************************************************415Ø
      DIMENSION Y(N),DY(N),YXE(N),INFO(15)                              416Ø
      DO 1Ø I=1,N                                                       417Ø
      YXE(I)=Y(I)+DY(I)/SIGMA                                           418Ø
   1Ø CONTINUE                                                          419Ø
      CALL F(N,YXE)                                                     42ØØ
      INFO(7)=INFO(7)+1                                                 421Ø
      ETAT=Ø.Ø                                                          422Ø
      ETAE=Ø.Ø                                                          423Ø
      DO 2Ø I=1,N                                                       424Ø
      ETAT=ETAT+Y(I)*Y(I)                                               425Ø
      E=YXE(I)-DY(I)                                                    426Ø
      ETAE=ETAE+E*E                                                     427Ø
   2Ø CONTINUE                                                          428Ø
      ETAT=TOL+TOL*SQRT(ETAT/FLOAT(N))                                  429Ø
      ETAE=SQRT(ETAE/FLOAT(N))/SIGMA+APR                                43ØØ
      H=SQRT(ETAT/ETAE)/SIGMA/1Ø.Ø                                      431Ø
C                                                                       432Ø
      RMMAX=INFO(11)                                                    433Ø
      BETA=(Ø.Ø3*RMMAX+Ø.44)*RMMAX*RMMAX                                434Ø
      IF(H.GT.BETA/SIGMA)H=BETA/SIGMA                                   435Ø
      RETURN                                                            436Ø
      END                                                               437Ø


      SUBROUTINE PARAM(C,LA,B,INFO)                                     438Ø
C*************************************************************************439Ø
C* PARAM DETERMINES THE PARAMETERS OF THE INTEGRATION SCHEME.THESE     * 44ØØ
C* PARAMETERS ARE EXPRESSIONS DEPENDING ON THE COEFFICIENTS OF THE     * 441Ø
C* STABILITY POLYNOMIALS,WHICH ARE STORED IN THE ARRAY D. DURING       * 442Ø
C* THE START,I.E.INFO(13) IS Ø OR 1,THE PARAMETERS OF A ONE-STEP SCHE- * 443Ø
C* ME ARE DETERMINED.                                                  * 444Ø
C*************************************************************************445Ø
      REAL LA                                                           446Ø
      DIMENSION C(12),LA(12),B(2),D(44Ø),P(13),S(13),INFO(15)           447Ø
      INTEGER ORDER                                                     448Ø
      DATA D(1),D(2),D(3),D(4),D(5),D(6),D(7),D(8),D(9),D(1Ø),          449Ø
     +D(11),D(12),D(13),D(14),D(15),D(16),D(17),D(18),D(19),D(2Ø),      45ØØ
     +D(21),D(22),D(23),D(24),D(25),D(26),D(27),D(28),D(29),D(3Ø)/      451Ø
     +.54545454545454E+ØØ,.3297422213950 3E+ØØ,.1587454Ø96372ØE-Ø1,     452Ø
     +.54545454545454E+ØØ,.3295777937240 4E+ØØ,.188Ø7742712872E-Ø1,     453Ø
     +.2693140629566 8E-Ø3,.54545454545454E+ØØ,.329596336269 66E+ØØ,    454Ø
     +.19843848141588E-Ø1,.3837Ø516974646E-Ø3,.232140Ø8199836E-Ø5,      455Ø
     +.54545454545454E+ØØ,.3294Ø48Ø78Ø399E+ØØ,.2Ø289622974884E-Ø1,     456Ø
     +.43922728369494E-Ø3,.38881393659393E-Ø5,.12Ø5863380 6519E-Ø7,    457Ø
     +.54545454545454E+ØØ,.3291573Ø747582E+ØØ,.2Ø529934599533E-Ø1,     458Ø
     +.47Ø14565Ø7Ø18ØE-Ø3,.4872995929Ø595E-Ø5,.23293337843875E-Ø7,     459Ø
     +.41768893290961E-1Ø,.54545454545454E+ØØ,.3294Ø29Ø556199E+ØØ,      46ØØ
     +.2Ø695785946957E-Ø1,.489593Ø52Ø8277E-Ø3,.5525Ø561672575E-Ø5/     461Ø
      DATA D(31),D(32),D(33),D(34),D(35),D(36),D(37),D(38),D(39),       462Ø
     +D(4Ø),D(41),D(42),D(43),D(44),D(45),D(46),D(47),D(48),D(49),      463Ø
```

```
+D(5Ø),D(51),D(52),D(53),D(54),D(55),D(56),D(57),D(58),D(59),          464Ø
+D(6Ø)/                                                               465Ø
+.32Ø4257286654ØE-Ø7,.92217187Ø87773E-1Ø,.1Ø43159677Ø258E-12,        466Ø
+.54545454545454E+ØØ,.329Ø4622Ø47855E+ØØ,.2Ø769127247467E-Ø1,        467Ø
+.5Ø1448731Ø8237E-Ø3,.59538312498493E-Ø5,.384136394Ø4624E-Ø7,        468Ø
+.13735161731261E-Ø9,.25579Ø38177Ø72E-12,.1935532554926ØE-15,        469Ø
+.54545454545454E+ØØ,.329Ø24Ø38254Ø4E+ØØ,.2Ø835887364795E-Ø1,        47ØØ
+.51Ø19371265378E-Ø3,.626712638488 34E-Ø5,.4327329421167ØE-Ø7,       471Ø
+.17557956622Ø83E-Ø9,.41529Ø99936815E-12,.5297776Ø638Ø1ØE-15,        472Ø
+.2816239662392Ø2E-18,.54545454545454E+ØØ,.329ØØ7Ø177Ø523E+ØØ,       473Ø
+.2Ø8477Ø9891773E-Ø1,.51474Ø22625718E-Ø3,.6465544445Ø446E-Ø5,        474Ø
+.46694681553560E-Ø7,.2Ø545752812528E-Ø9,.5598568365Ø846E-12/        475Ø
 DATA D(61),D(62),D(63),D(64),D(65),D(66),D(67),D(68),D(69),          476Ø
+D(7Ø),D(71),D(72),D(73),D(74),D(75),D(76),D(77),D(78),D(79),         477Ø
+D(8Ø),D(81),D(82),D(83),D(84),D(85),D(86),D(87),D(88),D(89),         478Ø
+D(9Ø)/                                                               479Ø
+.9223894Ø881933E-15,.8417148Ø799272E-18,.32655765Ø72494E-21,        48ØØ
+.54545454545454E+ØØ,.32923Ø47518706E+ØØ,.2Ø9421Ø551445ØE-Ø1,        481Ø
+.52155114179973E-Ø3,.666984Ø3229Ø1E-Ø5,.4978729Ø971999E-Ø7,        482Ø
+.231751Ø9Ø73Ø32E-Ø9,.6929Ø153772379E-12,.133Ø9584971259E-14,        483Ø
+.15876152476718E-17,.1Ø7Ø27944Ø9632E-2Ø,.31159779644428E-24,        484Ø
+.54545454545454E+ØØ,.32888824622955E+ØØ,.2Ø93Ø564Ø82556E-Ø1,        485Ø
+.52398Ø48369937E-Ø3,.6786531895 1772E-Ø5,.5189226338650 3E-Ø7,      486Ø
+.2516Ø762750795E-Ø9,.8Ø319349Ø35922E-12,.171Ø5594522426E-14,        487Ø
+.24Ø63258323529E-17,.21467888957759E-2Ø,.11ØØ2739569540E-23,        488Ø
+.24672233557657E-27,.45454545454545E+ØØ,.39753Ø5Ø587769E+ØØ/        489Ø
 DATA D(91),D(92),D(93),D(94),D(95),D(96),D(97),D(98),D(99),          49ØØ
+D(1ØØ),D(1Ø1),D(1Ø2),D(1Ø3),D(1Ø4),D(1Ø5),D(1Ø6),D(1Ø7),            491Ø
+D(1Ø8),D(1Ø9),D(11Ø),D(111),D(112),D(113),D(114),D(115),            492Ø
+D(116),D(117),D(118),D(119),D(12Ø)/                                 493Ø
+.191Ø6Ø34236683E-Ø1,.45454545454545E+ØØ,.39769493354868E+ØØ,       494Ø
+.226751ØØ9226Ø8E-Ø1,.32438614977749E-Ø3,.45454545454545E+ØØ,       495Ø
+.39767639 1ØØ3Ø6E+ØØ,.23921246939481E-Ø1,.46221334545758E-Ø3,     496Ø
+.279454925397 96E-Ø5,.45454545454545E+ØØ,.39786791946874E+ØØ,     497Ø
+.245Ø9754Ø44867E-Ø1,.53Ø71848Ø1Ø798E-Ø3,.47ØØ2589 4ØØ275E-Ø5,     498Ø
+.145853Ø3356181E-Ø7,.45454545454545E+ØØ,.39811541979691E+ØØ,      499Ø
+.24864589588956E-Ø1,.5699886Ø293388E-Ø3,.591383 4Ø6Ø351ØE-Ø5,     5ØØØ
+.283ØØ6Ø8926316E-Ø7,.5Ø812598486162E-1Ø,.45454545454545E+ØØ,      5Ø1Ø
+.39786982171Ø73E+ØØ,.25ØØØØ2282553E-Ø1,.59148858437864E-Ø3,      5Ø2Ø
+.66757562996758E-Ø5,.3872Ø46896477ØE-Ø7,.11144761163396E-Ø9/     5Ø3Ø
 DATA D(121),D(122),D(123),D(124),D(125),D(126),D(127),               5Ø4Ø
+D(128),D(129),D(13Ø),D(131),D(132),D(133),D(134),D(135),            5Ø5Ø
+D(136),D(137),D(138),D(139),D(14Ø),D(141),D(142),D(143),            5Ø6Ø
+D(144),D(145),D(146),D(147),D(148),D(149),D(15Ø)/                   5Ø7Ø
+.126Ø8153728ØØØE-12,.45454545454545E+ØØ,.3982265Ø679417E+ØØ,       5Ø8Ø
+.25183525Ø138Ø3E-Ø1,.6Ø88Ø314Ø13113E-Ø3,.72358433Ø14328E-Ø5,       5Ø9Ø
+.467253171ØØ1Ø6E-Ø7,.167195945Ø35Ø1E-Ø9,.31157527742621E-12,      51ØØ
+.2359Ø37Ø48211ØE-15,.45454545454545E+ØØ,.398248689Ø1869E+ØØ,      511Ø
+.25265819897767E-Ø1,.6191Ø411544859E-Ø3,.76Ø7471114885ØE-Ø5,      512Ø
+.525363441Ø1559E-Ø7,.21317642667569E-Ø9,.5Ø4213455ØØ4Ø6E-12,     513Ø
+.643176885 39Ø28E-15,.34187163161474E-18,.45454545454545E+ØØ,     514Ø
+.3982657Ø95675ØE+ØØ,.25287844Ø91655E-Ø1,.62494425734623E-Ø3,      515Ø
+.7854Ø29693845ØE-Ø5,.5674315145640 8E-Ø7,.24973869169769E-Ø9,     516Ø
+.68Ø666567167935E-12,.1121626Ø688Ø42E-14,.1Ø2368Ø2978533E-17/     517Ø
 DATA D(151),D(152),D(153),D(154),D(155),D(156),D(157),               518Ø
+D(158),D(159),D(16Ø),D(161),D(162),D(163),D(164),D(165),            519Ø
+D(166),D(167),D(168),D(169),D(17Ø),D(171),D(172),D(173),            52ØØ
+D(174),D(175),D(176)/                                              521Ø
+.3972Ø657964596E-21,.45454545454545E+ØØ,.398Ø42252Ø8567E+ØØ,       522Ø
+.253417Ø8Ø58167E-Ø1,.631515Ø1325496E-Ø3,.8Ø8Ø7554649258E-Ø5,       523Ø
+.6Ø3546644Ø2269E-Ø7,.2811164 2Ø92611E-Ø9,.841Ø6334638657E-12,     524Ø
+.1616728Ø397671E-14,.1929994178 99Ø7E-17,.13Ø21732956413E-2Ø,     525Ø
+.3794453664 7ØØE-24,.45454545454545E+ØØ,.39838448 1Ø4318E+ØØ,    526Ø
+.25416762495946E-Ø1,.63697712Ø76Ø76E-Ø3,.8255198 35Ø8228E-Ø5,    527Ø
+.6314972988877ØE-Ø7,.3Ø6298587Ø8888E-Ø9,.978Ø868191644ØE-12,     528Ø
+.2Ø836563 13Ø596E-14,.2932Ø679574331E-17,.26166497Ø172Ø4E-2Ø,     529Ø
+.13415328284784E-23,.3ØØ92796196223E-27/                          53ØØ
 DATA D(177),D(178),D(179),D(18Ø),D(181),D(182),D(183),D(184),       531Ø
+D(185),D(186),D(187),D(188),D(189),D(19Ø),D(191),D(192),D(193),     532Ø
+D(194),D(195),D(196),D(197),D(198),D(199),D(2ØØ),D(2Ø1),D(2Ø2),     533Ø
+D(2Ø3),D(2Ø4),D(2Ø5),D(2Ø6)/                                       534Ø
+-.58Ø6451612 9Ø32E+ØØ,-.21559395174672E+ØØ,-.3Ø544696579492E-Ø1,   535Ø
+-.58Ø6451612 9Ø32E+ØØ,-.19115223Ø31554E+ØØ,-.2947383478 61Ø2E-Ø1, 536Ø
+-.1ØØ66347258135E-Ø2,-.58Ø6451612 9Ø32E+ØØ,-.182333Ø7297138E+ØØ,  537Ø
+-.28236544473632E-Ø1,-.12Ø3ØØ396Ø1232E-Ø2,-.152Ø8ØØ8334815E-Ø4,  538Ø
+-.58Ø6451612 9Ø32E+ØØ,-.17833Ø69941496E+ØØ,-.28475246894827E-Ø1,  539Ø
```

```
     +-.146Ø63Ø2Ø3Ø482E-Ø2,-.299641113646ØØE-Ø4,-.213438Ø4115613E-Ø6,
     +-.58Ø64516129Ø32E+ØØ,-.1761Ø367Ø98315E+ØØ,-.2826Ø676645Ø9ØE-Ø1,
     +-.1532245881Ø336E-Ø2,-.3658632Ø114212E-Ø4,-.39867529427935E-Ø6,
     +-.16226665135199E-Ø8,-.58Ø64516129Ø32E+ØØ,-.1748339Ø114911E+ØØ,
     +-.27741186226246E-Ø1,-.14815956989918E-Ø2,-.363Ø1Ø45393729E-Ø4/
      DATA D(2Ø7),D(2Ø8),D(2Ø9),D(21Ø),D(211),D(212),D(213),D(214),
     +D(215),D(216),D(217),D(218),D(219),D(22Ø),D(221),D(222),D(223),
     +D(224),D(225),D(226),D(227),D(228),D(229),D(23Ø),D(231),D(232),
     +D(233),D(234),D(235),D(236)/
     +-.44685ØØ755Ø778E-Ø6,-.268755845Ø7281E-Ø8,-.62831231Ø83352E-11,
     +-.58Ø64516129Ø32E+ØØ,-.17396Ø71288671E+ØØ,-.28684134226593E-Ø1,
     +-.17432495919146E-Ø2,-.5Ø8458600Ø92993E-Ø4,-.79137773126678E-Ø6,
     +-.67374753547729E-Ø8,-.29589252318632E-1Ø,-.52416294Ø635Ø7E-13,
     +-.58Ø64516129Ø32E+ØØ,-.17339878373842E+ØØ,-.28Ø447272720Ø7E-Ø1,
     +-.16253143532453E-Ø2,-.4581ØØ65724231E-Ø4,-.71155788ØØ3856E-Ø6,
     +-.64Ø49Ø7274814ØE-Ø8,-.33259355915939E-1Ø,-.9239227819Ø522E-13,
     +-.1Ø625147968957E-15,-.58Ø64516129Ø32E+ØØ,-.17144173377ØØ9E+ØØ,
     +-.28Ø15Ø92938518E-Ø1,-.16157222373633E-Ø2,-.46227724829Ø57E-Ø4,
     +-.75479222345544E-Ø6,-.74894835Ø17511E-Ø8,-.4597905353 5837E-1Ø/
      DATA D(237),D(238),D(239),D(24Ø),D(241),D(242),D(243),D(244),
     +D(245),D(246),D(247),D(248),D(249),D(25Ø),D(251),D(252),D(253),
     +D(254),D(255),D(256),D(257),D(258),D(259),D(26Ø),D(261),D(262),
     +D(263),D(264),D(265),D(266)/
     +-.17Ø6Ø13779677ØE-12,-.35Ø56663Ø86597E-15,-.3Ø628886618191E-18,
     +-.58Ø64516129Ø32E+ØØ,-.173ØØ284166294E+ØØ,-.273280Ø9Ø24214E-Ø1,
     +-.15Ø56Ø21545Ø51E-Ø2,-.41367453292114E-Ø4,-.65884289780Ø74E-Ø6,
     +-.65536146317341E-Ø8,-.42Ø91Ø25159364E-1Ø,-.1747997705 6317E-12,
     +-.453684Ø682Ø998E-15,-.669287Ø32Ø8412E-18,-.42844311155752E-21,
     +-.58Ø64516129Ø32E+ØØ,-.1722936696Ø984E+ØØ,-.288ØØ272381574E-Ø1,
     +-.1859617443Ø42ØE-Ø2,-.59976978730906E-Ø4,-.11Ø9Ø566945359E-Ø5,
     +-.12745935692238E-Ø7,-.95158694657869E-1Ø,-.469694Ø6476975E-12,
     +-.15218Ø64136528E-14,-.3113Ø9521Ø85Ø8E-17,-.36466961532119E-2Ø,
     +-.18644934368193E-23,.158Ø64516129Ø3E+Ø1,.15Ø59165323919E+Ø1/
      DATA D(267),D(268),D(269),D(27Ø),D(271),D(272),D(273),D(274),
     +D(275),D(276),D(277),D(278),D(279),D(28Ø),D(281),D(282),D(283),
     +D(284),D(285),D(286),D(287),D(288),D(289),D(29Ø),D(291),D(292),
     +D(293),D(294),D(295),D(296)/
     +.16978945451Ø19E+ØØ,.158Ø64516129Ø3E+Ø1,.14814748109607E+Ø1,
     +.19316Ø31414798E+ØØ,.62334163398843E-Ø2,.158Ø64516129Ø3E+Ø1,
     +.14726556536165E+Ø1,.2ØØ74218117966E+ØØ,.86336976630508E-Ø2,
     +.11558Ø84587197E-Ø3,.158Ø64516129Ø3E+Ø1,.146865328ØØ6Ø1E+Ø1,
     +.2Ø498325715728E+ØØ,.99938118863291E-Ø2,.1992234803629 6E-Ø3,
     +.1391920144 8922E-Ø5,.158Ø64516129Ø3E+Ø1,.14664262516283E+Ø1,
     +.2Ø699571533935E+ØØ,.1Ø68Ø2754Ø5545E-Ø1,.249334Ø5Ø67263E-Ø3,
     +.26855Ø39111666E-Ø5,.1Ø85485Ø7136Ø1E-Ø7,.158Ø64516129Ø3E+Ø1,
     +.14651564817943E+Ø1,.2Ø774599475456E+ØØ,.1Ø971861Ø52267E-Ø1,
     +.275243688300Ø2E-Ø3,.354Ø3656934423E-Ø5,.22575321236761E-Ø7/
      DATA D(297),D(298),D(299),D(3ØØ),D(3Ø1),D(3Ø2),D(3Ø3),D(3Ø4),
     +D(3Ø5),D(3Ø6),D(3Ø7),D(3Ø8),D(3Ø9),D(31Ø),D(311),D(312),D(313),
     +D(314),D(315),D(316),D(317),D(318),D(319),D(32Ø),D(321),D(322),
     +D(323),D(324),D(325),D(326)/
     +.56574487882585E-1Ø,.158Ø64516129Ø3E+Ø1,.14642832935319E+Ø1,
     +.2Ø95621310173ØE+ØØ,.115Ø9566107375E-Ø1,.31Ø97988163828E-Ø3,
     +.4563Ø9861333Ø2E-Ø5,.37Ø79641130883E-Ø7,.1568259Ø95Ø194E-Ø9,
     +.269334300355 88E-12,.158Ø64516129Ø3E+Ø1,.14637213643836E+Ø1,
     +.2Ø9484653211Ø1E+ØØ,.11536416747709E-Ø1,.3178461487Ø548E-Ø3,
     +.491167200470Ø9E-Ø5,.445281960804 81E-Ø7,.235Ø5328331393E-Ø9,
     +.66870118493809E-12,.79239438466385E-15,.158Ø64516129Ø3E+Ø1,
     +.14617643144152E+Ø1,.2114120688458 4E+ØØ,.1180505628802 4E-Ø1,
     +.3344Ø743994479E-Ø3,.5441423529388ØE-Ø5,.53918Ø86434111E-Ø7,
     +.33Ø7545Ø1919Ø3E-Ø9,.12264Ø57386754E-11,.25181Ø64036724E-14/
      DATA D(327),D(328),D(329),D(33Ø),D(331),D(332),D(333),D(334),
     +D(335),D(336),D(337),D(338),D(339),D(34Ø),D(341),D(342),D(343),
     +D(344),D(345),D(346),D(347),D(348),D(349),D(35Ø),D(351),D(352)/
     +.219776160728 1ØE-17,.158Ø64516129Ø3E+Ø1,.14633254223Ø81E+Ø1,
     +.2Ø9163877Ø3869E+ØØ,.1157945455Ø239E-Ø1,.32857305529ØØ8E-Ø3,
     +.54458388Ø15784E-Ø5,.56371977519294E-Ø7,.3754508369244ØE-Ø9,
     +.16Ø91ØØ595Ø325E-11,.42884398776166E-14,.64665832685414E-17,
     +.42148457924105E-2Ø,.158Ø64516129Ø3E+Ø1,.14626162502550E+Ø1,
     +.21134531244915E+ØØ,.121Ø8121568554E-Ø1,.35894153745450E-Ø3,
     +.62664422624671E-Ø5,.69193362616297E-Ø7,.5Ø195179261835E-Ø9,
     +.24253292Ø36317E-11,.7731Ø3654618Ø3E-14,.15613921810526E-16,
     +.181Ø2819599155E-19,.91776107476727E-23/
C
      DATA D(353),D(354),D(355),D(356),D(357),D(358),D(359),D(36Ø),
     +D(361),D(362),D(363),D(364),D(365),D(366),D(367),D(368),D(369),
```

```
     +D(37Ø),D(371),D(372),D(373),D(374),D(375),D(376),D(377),D(378),     616Ø
     +D(379),D(38Ø),D(381),D(382),D(383),D(384),D(385),D(386),D(387),     617Ø
     +D(388),D(389),D(39Ø),D(391),D(392),D(393),D(394),D(395),D(396),     618Ø
     +D(397),D(398),D(399),D(4ØØ),D(4Ø1),D(4Ø2)/                          619Ø
     +1.,1.,.5,                                                           62ØØ
     +1.,1.,Ø.5,633Ø4Ø85.E-Ø9,1.,1.,Ø.5,79Ø27358.E-Ø9,37Ø89254.E-1Ø,      621Ø
     +1.,1.,Ø.5,856Ø5575.E-Ø9,56773923.E-1Ø,12758716.E-11,1.,1.,Ø.5,      622Ø
     +89Ø18496.E-Ø9,6794ØØ3.E-1Ø,23143226.E-11,2898388.E-12,1.,1.,Ø.5,    623Ø
     +91Ø254Ø8.E-Ø9,74822425.E-1Ø,3Ø56758Ø.E-11,6Ø72ØØ1.E-12,             624Ø
     +4679323.E-14,1.,1.,Ø.5,923Ø8224.E-Ø9,79335426.E-1Ø,35849723.E-11,   625Ø
     +88ØØ161.E-12,111Ø8474.E-14,5648779.E-16,1.,1.,Ø.5,93178948.E-Ø9,    626Ø
     +82451157.E-1Ø,3968Ø345.E-11,11ØØØ3Ø1.E-12,17552367.E-14/            627Ø
      DATA D(4Ø3),D(4Ø4),D(4Ø5),D(4Ø6),D(4Ø7),D(4Ø8),D(4Ø9),D(41Ø),       628Ø
     +D(411),D(412),D(413),D(414),D(415),D(416),D(417),D(418),D(419),     629Ø
     +D(42Ø),D(421),D(422),D(423),D(424),D(425),D(426),D(427),D(428),     63ØØ
     +D(429),D(43Ø),D(431),D(432),D(433),D(434),D(435),D(436),D(437),     631Ø
     +D(438),D(439),D(44Ø)/                                               632Ø
     +14976734.E-16,5293311.E-18,1.,1.,Ø.5,93797465.E-Ø9,8469Ø176.E-1Ø,   633Ø
     +42524183.E-11,12746945.E-12,23355Ø62.E-14,25642831.E-16,            634Ø
     +15495967.E-18,3962Ø8.E-2Ø,1.,1.,Ø.5,94252811.E-Ø9,86352191.E-1Ø,    635Ø
     +446838Ø2.E-11,1413517Ø.E-12,28359Ø79.E-14,36242Ø82.E-16,            636Ø
     +28591262.E-18,12691526.E-2Ø,24249737.E-23,1.,1.,Ø.5,94597848.E-Ø9,  637Ø
     +87619296.E-1Ø,46357872.E-11,1524691Ø.E-12,32599754.E-14,            638Ø
     +461Ø7927.E-16,42819928.E-18,2511Ø371.E-2Ø,84319465.E-23,             639Ø
     +123571Ø5.E-25/                                                      64ØØ
C                                                                         641Ø
      ORDER=INFO(12)                                                      642Ø
      M=INFO(9)                                                           643Ø
      IF(INFO(13).LT.2) GOTO 5Ø                                           644Ø
      M1=176*(ORDER-1)+M*(M+1)/2-3                                        645Ø
      M2=88+M1                                                            646Ø
      MM=M+1                                                              647Ø
      DO 1Ø J=1,MM                                                        648Ø
      M1PJ=M1+J                                                           649Ø
      M2PJ=M2+J                                                           65ØØ
      P(J)=D(M1PJ)                                                        651Ø
      S(J)=D(M2PJ)                                                        652Ø
   1Ø CONTINUE                                                            653Ø
      IF(M.GT.2) GOTO 2Ø                                                  654Ø
      P(4)=Ø.Ø                                                            655Ø
      S(4)=Ø.Ø                                                            656Ø
   2Ø DD=1.375-.6*FLOAT(ORDER-1)                                          657Ø
      E=P(2)+2.Ø*(-P(3)+P(4)+S(4))                                        658Ø
      C(M)=(E/DD-((DD-Ø.5)/DD)**2)/(2.+E)                                 659Ø
      LA(M)=1.Ø/DD-C(M)                                                   66ØØ
      LA(M-1)=S(3)/LA(M)                                                  661Ø
      C(M-1)=P(3)/LA(M)                                                   662Ø
      B(1)=(P(2)-C(M))/LA(M)                                              663Ø
      B(2)=P(1)                                                           664Ø
      IF(M.GT.2) GOTO 3Ø                                                  665Ø
      RETURN                                                              666Ø
C                                                                         667Ø
   3Ø MM=M-2                                                              668Ø
      DO 4Ø J=1,MM                                                        669Ø
      MP2MJ=M+2-J                                                         67ØØ
      MP1MJ=M+1-J                                                         671Ø
      C(J)=P(MP2MJ)/S(MP1MJ)                                              672Ø
      LA(J)=S(MP2MJ)/S(MP1MJ)                                             673Ø
   4Ø CONTINUE                                                            674Ø
      RETURN                                                              675Ø
C                                                                         676Ø
   5Ø M1=352+M*(M+1)/2-3                                                  677Ø
      MM=M+1                                                              678Ø
      DO 6Ø J=1,MM                                                        679Ø
      M1PJ=M1+J                                                           68ØØ
   6Ø S(J)=D(M1PJ)                                                        681Ø
      B(1)=Ø.Ø                                                            682Ø
      B(2)=Ø.Ø                                                            683Ø
      C(M)=Ø.Ø                                                            684Ø
      LA(M)=S(1)                                                          685Ø
      MM=M-1                                                              686Ø
      DO 7Ø J=1,MM                                                        687Ø
      MP2MJ=M+2-J                                                         688Ø
      MP1MJ=M+1-J                                                         689Ø
      LA(J)=S(MP2MJ)/S(MP1MJ)                                             69ØØ
      C(J)=Ø.Ø                                                            691Ø
```

```
   7Ø CONTINUE                                                          692Ø
      RETURN                                                            693Ø
      END                                                               694Ø


      SUBROUTINE POWERM(N,Y,Y1,YXE,DY,DY1,F,SIGMA,APR,IFLAG,INFO)       695Ø
C*********************************************************************  696Ø
C* IF ON ENTRY SIGMA(1)=Ø POWERM COMPUTES AN ESTIMATION OF THE SPECTR- *  697Ø
C* AL RADIUS OF THE JACOBIAN BY MEANS OF AN ADJUSTED POWER METHOD.THE  *  698Ø
C* ITERATION IS STOPPED IF TWO SUCCESSIVE ITERATES DIFFER RELATIVELY   *  699Ø
C* LESS THAN Ø.ØØ1.THE MINIMAL NUMBER OF ITERATIONS IS 5.IF THE COMPU- *  7ØØØ
C* TATION DID NOT SUCCEED WITHIN 5Ø ITERATIONS AN ERRORMESSAGE IS GIV- *  7Ø1Ø
C* EN.AS A SAFETY MARGIN,THE LAST ITERATE IS ENLARGED WITH 1Ø PERCENT. *  7Ø2Ø
C* THE RESULT IS STORED IN SIGMA(1).                                   *  7Ø3Ø
C*                                                                     *  7Ø4Ø
C* IF ON ENTRY SIGMA(1) NOT EQUALS Ø POWERM PERFORMS THREE ITERATIONS  *  7Ø5Ø
C* WITH THE ADJUSTED POWER METHOD.IF THE THIRD ITERATE IS MORE THAN 1Ø *  7Ø6Ø
C* PERCENT SMALLER THAN SIGMA(2),THE ITERATION IS CONTINUED AS IF      *  7Ø7Ø
C* SIGMA(1)=Ø.IN THIS CASE THE THIRD ITERATE IS STORED IN SIGMA(2).    *  7Ø8Ø
C*                                                                     *  7Ø9Ø
C* THE POWER METHOD REQUIRES THREE WORK ARRAYS.WE USE YXE,DY AND DY1,  *  71ØØ
C* WHERE DY AND DY1 ARE OVERWRITTEN.ON ENTRY OF THIS ROUTINE DY AND    *  711Ø
C* DY1 CONTAIN F(Y) AT Y=Y(X) AND Y=Y(X-H),RESPECTIVELY.THUS ON EXIT   *  712Ø
C* OF THIS ROUTINE TWO EXTRA EVALUATIONS OF F ARE NECESSARY FOR RESTO- *  713Ø
C* RING THE DERIVATIVES.                                               *  714Ø
C*                                                                     *  715Ø
C* THIS CODE USES THE CDC SYSTEM SUBPROGRAMS RANSET AND RANF,GENE-     *  716Ø
C* RATING RANDOM VALUES FROM THE INTERVAL IØ,1 . THE ARGUMENT OF RANF  *  717Ø
C* IS DUMMY AND IGNORED.RANSET INITIALIZES THE GENERATIVE VALUE OF RANF*  718Ø
C*********************************************************************  719Ø
      DIMENSION Y(N),Y1(N),YXE(N),DY(N),DY1(N),SIGMA(2),INFO(15)        72ØØ
      REAL NORM,NORMØ                                                   721Ø
      IF(INFO(2).EQ.3) INFO(2)=1                                        722Ø
      INFO(15)=Ø                                                        723Ø
      TOLLIP=1.E+4*APR                                                  724Ø
      SIGM=Ø.Ø                                                          725Ø
      SØ=Ø.Ø                                                            726Ø
      CALL RANSET(Ø)                                                    727Ø
      DO 1Ø I=1,N                                                       728Ø
      RA=2.*RANF(IDUM)-1.                                               729Ø
      YXE(I)=DY(I)                                                      73ØØ
      IF(Y(I).EQ.Ø.Ø) DY(I)=RA*TOLLIP                                   731Ø
      IF(Y(I).NE.Ø.Ø) DY(I)=Y(I)*(1.Ø+TOLLIP*RA)                        732Ø
      DY1(I)=DY(I)                                                      733Ø
      SØ=SØ+DY(I)*DY(I)                                                 734Ø
   1Ø CONTINUE                                                          735Ø
      NORMØ=TOLLIP*SQRT(SØ)                                             736Ø
      IF(NORMØ.LT.TOLLIP)NORMØ=TOLLIP                                   737Ø
      CALL F(N,DY1)                                                     738Ø
      INFO(7)=INFO(7)+1                                                 739Ø
      INFO(8)=INFO(8)+1                                                 74ØØ
C                                                                       741Ø
      DO 7Ø K=1,51                                                      742Ø
      IF(K.LT.51) GOTO 2Ø                                               743Ø
      IFLAG=3                                                           744Ø
      RETURN                                                            745Ø
C                                                                       746Ø
   2Ø SØ=Ø                                                              747Ø
      DO 3Ø I=1,N                                                       748Ø
      S1=YXE(I)-DY1(I)                                                  749Ø
      SØ=SØ+S1*S1                                                       75ØØ
   3Ø CONTINUE                                                          751Ø
      NORM=SQRT(SØ)                                                     752Ø
      SIGM1=SIGM                                                        753Ø
      SIGM=NORM/NORMØ                                                   754Ø
C                                                                       755Ø
      IF(K.EQ.3.AND.SIGMA(1).EQ.Ø.Ø) SIGMA(2)=SIGM                      756Ø
      IF(K.LE.2.OR.SIGMA(1).EQ.Ø.Ø) GOTO 4Ø                             757Ø
      IF(SIGM.GE.Ø.9*SIGMA(2)) GOTO 8Ø                                  758Ø
      SIGMA(2)=SIGM                                                     759Ø
      SIGMA(1)=Ø.                                                       76ØØ
C                                                                       761Ø
   4Ø IF(ABS(SIGM1-SIGM)/SIGM.GT.Ø.ØØ1.OR.K.LE.4) GOTO 5Ø               762Ø
      SIGMA(1)=1.1*SIGM                                                 763Ø
      GOTO 8Ø                                                           764Ø
   5Ø DO 6Ø I=1,N                                                       765Ø
```

```
   6Ø YXE(I)=DY(I)+(YXE(I)-DY1(I))/SIGM                                  766Ø
      CALL F(N,YXE)                                                      767Ø
      INFO(7)=INFO(7)+1                                                  768Ø
      INFO(8)=INFO(8)+1                                                  769Ø
   7Ø CONTINUE                                                           77ØØ
C                                                                        771Ø
   8Ø DO 9Ø I=1,N                                                        772Ø
   9Ø DY(I)=Y(I)                                                         773Ø
      CALL F(N,DY)                                                       774Ø
      INFO(7)=INFO(7)+1                                                  775Ø
      INFO(8)=INFO(8)+1                                                  776Ø
      IF(INFO(13).NE.Ø) GOTO 1ØØ                                         777Ø
      RETURN                                                             778Ø
  1ØØ DO 11Ø I=1,N                                                       779Ø
  11Ø DY1(I)=Y1(I)                                                       78ØØ
      CALL F(N,DY1)                                                      781Ø
      INFO(7)=INFO(7)+1                                                  782Ø
      INFO(8)=INFO(8)+1                                                  783Ø
      RETURN                                                             784Ø
      END                                                                785Ø


      SUBROUTINE MAXDEG(TOL,APR,IFLAG,INFO)                              786Ø
C*********************************************************************  787Ø
C* IN MAXDEG THE MAXIMAL DEGREES WITH RESPECT TO THE INTERNAL STABILI- *  788Ø
C* TY CONDITION ARE COMPUTED. IF ONE OF THESE MAXIMAL DEGREES FALLS    *  789Ø
C* OUTSIDE THE RANGE,AN ERRORMESSAGE IS GIVEN.                         *  79ØØ
C*********************************************************************  791Ø
      DIMENSION Q(11),INFO(15)                                           792Ø
      DATA Q(1),Q(2),Q(3),Q(4),Q(5),Q(6),Q(7),Q(8),Q(9),Q(1Ø),Q(11)/     793Ø
     +3.E1,1.E2,7.E2,4.E3,3.E4,2.E5,9.E5,5.E6,3.E7,2.E8,1.E9/            794Ø
      E=TOL/APR                                                          795Ø
      DO 1Ø I=2,12                                                       796Ø
      J=13-I                                                             797Ø
      IF(Q(J).LE.E) GOTO 2Ø                                              798Ø
   1Ø CONTINUE                                                           799Ø
      IFLAG=2                                                            8ØØØ
      RETURN                                                             8Ø1Ø
C                                                                        8Ø2Ø
   2Ø INFO(1Ø)=14-I                                                      8Ø3Ø
      DO 3Ø I=2,12                                                       8Ø4Ø
      J=13-I                                                             8Ø5Ø
      IF(1ØØ.Ø*Q(J).LE.E) GOTO 4Ø                                        8Ø6Ø
   3Ø CONTINUE                                                           8Ø7Ø
      IFLAG=2                                                            8Ø8Ø
      RETURN                                                             8Ø9Ø
C                                                                        81ØØ
   4Ø INFO(11)=14-I                                                      811Ø
      RETURN                                                             812Ø
      END                                                                813Ø


      SUBROUTINE MINDEG(H,SIGMA,INFO)                                    814Ø
C*********************************************************************  815Ø
C* MINDEG DETERMINES THE MINIMAL DEGREE M WHICH STILL GIVES RISE TO A  *  816Ø
C* STABLE INTEGRATION STEP.                                            *  817Ø
C*********************************************************************  818Ø
      DIMENSION INFO(15)                                                 819Ø
      LOGICAL START                                                      82ØØ
      L=INFO(12)+9                                                       821Ø
      MMAX=INFO(L)                                                       822Ø
      START=INFO(13).LT.2                                                823Ø
      IF(.NOT.START) BETAC=5.15+2.86*FLOAT(1-INFO(12))                   824Ø
      DO 1Ø M=2,MMAX                                                     825Ø
      RM=M                                                               826Ø
      IF(START) BETAC=Ø.Ø3*RM+Ø.44                                       827Ø
      IF(H.LE.BETAC*RM*RM/SIGMA) GOTO 2Ø                                 828Ø
   1Ø CONTINUE                                                           829Ø
      M=MMAX                                                             83ØØ
   2Ø INFO(9)=M                                                          831Ø
      RETURN                                                             832Ø
      END                                                                833Ø


      SUBROUTINE STEP(N,Y,Y1,Y2,YXE,DY,DY1,H,F,C,LA,B,INFO)              834Ø
C*********************************************************************  835Ø
```

```
C* STEP CONTAINS THE ACTUAL INTEGRATOR.FOR CONVENIENCE,THE ONE-STEP    *  836Ø
C* SCHEME IS ALSO FORMULATED AS A THREE-STEP SCHEME BY INTRODUCING ZE- *  837Ø
C* RO-PARAMETERS.ON EXIT OF THIS ROUTINE THE NEW CALCULATED SOLUTION   *  838Ø
C* VECTOR IS CONTAINED IN YXE.                                         *  839Ø
C*********************************************************************  84ØØ
      REAL LA                                                             841Ø
      DIMENSION Y(N),Y1(N),Y2(N),YXE(N),DY(N),DY1(N),                     842Ø
     +LA(12),C(12),B(2),INFO(15)                                          843Ø
      M=INFO(9)                                                           844Ø
      D=1.375-.6*FLOAT(INFO(12)-1)                                        845Ø
      IF(INFO(13).LT.2) D=1.Ø                                             846Ø
      DO 1Ø I=1,N                                                         847Ø
   1Ø YXE(I)=DY(I)                                                        848Ø
      IF(M.EQ.2) GOTO 4Ø                                                  849Ø
      MM=M-2                                                              85ØØ
C                                                                         851Ø
      DO 3Ø J=1,MM                                                        852Ø
      HC=H*C(J)                                                           853Ø
      HLA=H*LA(J)                                                         854Ø
      DO 2Ø I=1,N                                                         855Ø
      YXE(I)=Y(I)+HC*DY1(I)+HLA*YXE(I)                                    856Ø
   2Ø CONTINUE                                                            857Ø
      CALL F(N,YXE)                                                       858Ø
      INFO(7)=INFO(7)+1                                                   859Ø
   3Ø CONTINUE                                                            86ØØ
C                                                                         861Ø
   4Ø BM1=B(1)                                                            862Ø
      BM11=1.Ø-BM1                                                        863Ø
      HC=H*C(M-1)                                                         864Ø
      HLA=H*LA(M-1)                                                       865Ø
      DO 5Ø I=1,N                                                         866Ø
   5Ø YXE(I)=BM11*Y(I)+BM1*Y1(I)+HC*DY1(I)+HLA*YXE(I)                     867Ø
      D1=1.-D                                                             868Ø
      BM1=B(2)*D                                                          869Ø
      BM11=(1.-B(2))*D                                                    87ØØ
      HC=H*C(M)*D                                                         871Ø
      HLA=H*LA(M)*D                                                       872Ø
      CALL F(N,YXE)                                                       873Ø
      INFO(7)=INFO(7)+1                                                   874Ø
      DO 6Ø I=1,N                                                         875Ø
   6Ø YXE(I)=BM11*Y(I)+BM1*Y1(I)+D1*Y2(I)+HC*DY1(I)+HLA*YXE(I)            876Ø
      RETURN                                                              877Ø
      END                                                                 878Ø


      SUBROUTINE ESTIMA(N,Y,Y1,Y2,YXE,TOL,EPS,ERROR,INFO)                 879Ø
C*********************************************************************  88ØØ
C* ESTIMA CALCULATES THE LOCAL ERROR BOUND EPS=(1+NORM(Y))*TOL FOR THE *  881Ø
C* MIXED ERROR TEST AND ESTIMATES THE LOCAL ERROR ERROR.               *  882Ø
C*********************************************************************  883Ø
      DIMENSION Y(N),Y1(N),Y2(N),YXE(N),INFO(15),CONST(2)                 884Ø
      INTEGER ORDER                                                       885Ø
      CONST(1)=4.7Ø                                                       886Ø
      CONST(2)=Ø.79                                                       887Ø
      ORDER=INFO(12)                                                      888Ø
      EPS=Ø.Ø                                                             889Ø
      ERROR=Ø.Ø                                                           89ØØ
      IF(ORDER.EQ.2) GOTO 2Ø                                              891Ø
      DO 1Ø I=1,N                                                         892Ø
      YI=Y(I)                                                             893Ø
      EPS=EPS+YI*YI                                                       894Ø
      E=Y1(I)-YI-YI+YXE(I)                                                895Ø
      ERROR=ERROR+E*E                                                     896Ø
   1Ø CONTINUE                                                            897Ø
C                                                                         898Ø
      GOTO 4Ø                                                             899Ø
   2Ø DO 3Ø I=1,N                                                         9ØØØ
      EPS=EPS+Y(I)*Y(I)                                                   9Ø1Ø
      E=-Y2(I)+3.Ø*(Y1(I)-Y(I))+YXE(I)                                    9Ø2Ø
      ERROR=ERROR+E*E                                                     9Ø3Ø
   3Ø CONTINUE                                                            9Ø4Ø
C                                                                         9Ø5Ø
   4Ø EPS=TOL+TOL*SQRT(EPS/FLOAT(N))                                      9Ø6Ø
      ERROR=CONST(ORDER)*SQRT(ERROR/FLOAT(N))                             9Ø7Ø
      RETURN                                                              9Ø8Ø
      END                                                                 9Ø9Ø
```

```
      SUBROUTINE NEWH(EPSERR,HOLD,H,ALFA,HMAX,INFO)                    91ØØ
C*********************************************************************  911Ø
C* NEWH DELIVERS A NEW STEPLENGTH AND THE FACTOR ALFA BY WHICH THE     *  912Ø
C* STEPLENGTH IS CHANGED. EPSERR DENOTES EPS/ERROR.                    *  913Ø
C*********************************************************************  914Ø
      DIMENSION HMAX(2),INFO(15)                                       915Ø
      INTEGER ORDER                                                    916Ø
      ALFA=1.Ø                                                         917Ø
      IF(INFO(14).GE.3.OR.EPSERR.LE.1.Ø) GOTO 1Ø                       918Ø
      RETURN                                                           919Ø
C                                                                      92ØØ
   1Ø ORDER=INFO(12)                                                   921Ø
      ALFA=EPSERR**(1./FLOAT(ORDER+1))/(2.Ø-FLOAT(ORDER-1)*Ø.4)        922Ø
      IF(ALFA.GT.Ø.9.AND.ALFA.LT.1.1) ALFA=1.Ø                         923Ø
      IF(ALFA.NE.1.Ø) GOTO 2Ø                                          924Ø
C                                                                      925Ø
      RETURN                                                           926Ø
   2Ø IF(ALFA.GT.3.Ø) ALFA=3.Ø                                         927Ø
      IF(ALFA.LT.Ø.1) ALFA=Ø.1                                         928Ø
      H=HOLD*ALFA                                                      929Ø
      IF(H.GT.HMAX(ORDER)) H=HMAX(ORDER)                               93ØØ
      ALFA=H/HOLD                                                      931Ø
      RETURN                                                           932Ø
      END                                                              933Ø


      SUBROUTINE INTER1(N,Y,Y1,Y2,DY1,F,ALFA,INFO)                     934Ø
C*********************************************************************  935Ø
C* INTER1 COMPUTES VALUES FOR Y(X-ALFA*H) AND Y(X-2*ALFA*H) FROM Y(X), *  936Ø
C* Y(X-H) AND Y(X-2H) BY QUADRATIC INTERPOLATION,AND COMPUTES THE DER- *  937Ø
C* IVATIVE AT X-H BY CALLING F.                                        *  938Ø
C*********************************************************************  939Ø
      DIMENSION Y(N),Y1(N),Y2(N),DY1(N),INFO(15)                       94ØØ
      REAL NU                                                          941Ø
      NU=2.-ALFA                                                       942Ø
      C12=(NU-1.)*(NU-2.)/2.                                           943Ø
      C11=NU*(2.-NU)                                                   944Ø
      C1Ø=NU*(NU-1.)/2.                                                945Ø
      NU=2.-2.*ALFA                                                    946Ø
      C22=(NU-1.)*(NU-2.)/2.                                           947Ø
      C21=NU*(2.-NU)                                                   948Ø
      C2Ø=NU*(NU-1.)/2.                                                949Ø
C                                                                      95ØØ
      DO 1Ø I=1,N                                                      951Ø
      CYØ=Y(I)                                                         952Ø
      CY1=Y1(I)                                                        953Ø
      CY2=Y2(I)                                                        954Ø
      Y1(I)=C12*CY2+C11*CY1+C1Ø*CYØ                                    955Ø
      DY1(I)=Y1(I)                                                     956Ø
      Y2(I)=C22*CY2+C21*CY1+C2Ø*CYØ                                    957Ø
   1Ø CONTINUE                                                         958Ø
      CALL F(N,DY1)                                                    959Ø
      INFO(7)=INFO(7)+1                                                96ØØ
      INFO(14)=Ø                                                       961Ø
      RETURN                                                           962Ø
      END                                                              963Ø


     SUBROUTINE INTER2(N,Y,Y1,Y2,YXE,A)                                964Ø
C*********************************************************************  965Ø
C* INTER2 COMPUTES THE SOLUTION AT THE OUTPUT POINT XE=X-A*H BY QUAD-  *  966Ø
C* RATIC INTERPOLATION BETWEEN Y(X),Y(X-H) AND Y(X-2H).THE RESULT IS   *  967Ø
C* STORED IN YXE.                                                      *  968Ø
C*********************************************************************  969Ø
      DIMENSION Y(N),Y1(N),Y2(N),YXE(N)                                97ØØ
      REAL NU                                                          971Ø
      NU=2.-A                                                          972Ø
      C12=(NU-1.)*(NU-2.)/2.                                           973Ø
      C11=NU*(2.-NU)                                                   974Ø
      C1Ø=NU*(NU-1.)/2.                                                975Ø
      DO 1Ø I=1,N                                                      976Ø
   1Ø YXE(I)=C12*Y2(I)+C11*Y1(I)+C1Ø*Y(I)                              977Ø
      RETURN                                                           978Ø
      END                                                              979Ø
```

```
      SUBROUTINE SHIFT(N,Y,Y1,Y2,YXE,DY,DY1,X,H,F,INFO)                  98ØØ
C*********************************************************************  981Ø
C* SHIFT SHIFTS THE X AND Y VARIABLES AND COMPUTES F(Y(X+H))          *  982Ø
C*********************************************************************  983Ø
      DIMENSION Y(N),Y1(N),Y2(N),YXE(N),DY(N),DY1(N),INFO(15)            984Ø
      DO 1Ø I=1,N                                                        985Ø
      Y2(I)=Y1(I)                                                        986Ø
      Y1(I)=Y(I)                                                         987Ø
      Y(I)=YXE(I)                                                        988Ø
      DY1(I)=DY(I)                                                       989Ø
      DY(I)=Y(I)                                                         99ØØ
   1Ø CONTINUE                                                           991Ø
      CALL F(N,DY)                                                       992Ø
      INFO(7)=INFO(7)+1                                                  993Ø
      X=X+H                                                              994Ø
      RETURN                                                             995Ø
      END                                                                996Ø
```

ALGORITHM 554

BRENTM, A Fortran Subroutine for the Numerical Solution of Systems of Nonlinear Equations [C5]

JORGE J. MORÉ
Argonne National Laboratory
and
MICHEL Y. COSNARD
Université Scientifique et Médicale de Grenoble

Key Words and Phrases: nonlinear equations, numerical solution, Brent's method
CR Categories: 4.6, 5.15
Language: Fortran

1. DESCRIPTION

BRENTM is a subroutine designed to solve a system of n nonlinear equations in n variables by using a modification of Brent's method [1]. This modification, and numerical results for our implementation, are discussed in [2].

Our subroutine does not use any techniques which attempt to obtain global convergence, and therefore convergence is only guaranteed if the initial estimate for the solution is close enough. On the other hand, our code does seem to have a large region of convergence; convergence only occurs at a zero of the function, and if the iteration is not making satisfactory progress, then BRENTM will attempt to diagnose this situation and stop the iteration with an appropriate message.

The user is only required to provide a subroutine which calculates components of the function. Note that because of the nature of Brent's method, it is advantageous to arrange the components of the function in increasing order of nonlinearity; in particular, any linear components should appear first.

The accuracy of BRENTM is controlled by the parameters FTOL and XTOL. Convergence occurs if all residuals are at most FTOL in magnitude. Thus FTOL should be chosen with care since an appropriate value depends on the scaling of the functions. Convergence also occurs if BRENTM estimates that

$$\| x - x^* \|_\infty \leq \mathrm{TOL} \| x^* \|_\infty$$

where x^* is a solution to the system of nonlinear equations and x is the current estimate of x^*. Note that if x^* has components of widely different magnitude,

Received 7 February 1978; revised 27 August 1979; accepted 11 December 1979.

This work was performed under the auspices of the U.S. Department of Energy.
Authors' addresses: J.J. Moré, Applied Mathematics Division, Argonne National Laboratory, Argonne, IL 60439; M.Y. Cosnard, Mathematiques Appliquées-Informatique, Université Scientifique et Médicale de Grenoble, Boite Postal 53, 38041 Grenoble Cédex, France.

then there is a danger that the smaller components of x will have large relative errors, but this usually does not happen because of the fast convergence of the iteration.

2. BRENT1

The complete generality provided by BRENTM is usually not required, and in these cases, the calling sequence can be shortened and simplified. We have incorporated the following simplifications into the easy-to-use driver BRENT1. Note that BRENT1 has 8 parameters in the calling sequence, while BRENTM has 15.

(1) Only allow one tolerance TOL by setting FTOL and XTOL equal to TOL. Unless there is information about the scaling of the function, this is a natural choice.

(2) Let MOPT maximize $\log(k + 1)/(n + 2k + 1)$ for $k = 1, 2, \ldots, n$. This is the value recommended in [1] and [2].

(3) Set MAXFEV to $50(n + 3)$. This limit permits 100 iterations of the algorithm, and in our experience BRENTM has either succeeded or reported unsatisfactory progress before this limit is reached.

(4) Eliminate NFEV from the calling sequence. Many users are not interested in the number of function evaluations required by BRENTM, and in any case NFEV can be computed by the user through FCN.

(5) Condense all arrays not required by FCN into one work array WA of length LWA. BRENTM would need LWA to be at least $n(n + 3)$.

REFERENCES

1. BRENT, R.P. Some efficient algorithms for solving systems of nonlinear equations. *SIAM J. Numer. Anal. 10* (1973), 327-344.
2. MORÉ, J.J., AND COSNARD, M.Y. Numerical solution of nonlinear equations. *ACM Trans. Math. Softw. 5*, 1 (March 1979), 64-85.

ALGORITHM

```
C     **********                                                    ØØØØØØ1Ø
C                                                                   ØØØØØØ2Ø
C     THIS IS A SAMPLE PROGRAM FOR THE EASY-TO-USE VERSION BRENT1   ØØØØØØ3Ø
C     OF SUBROUTINE BRENTM. THIS PROGRAM SOLVES THE DISCRETE        ØØØØØØ4Ø
C     BOUNDARY VALUE PROBLEM DEFINED BY THE SYSTEM OF NONLINEAR     ØØØØØØ5Ø
C     EQUATIONS                                                     ØØØØØØ6Ø
C                                                                   ØØØØØØ7Ø
C     2*X(I) - X(I-1) - X(I+1)                                      ØØØØØØ8Ø
C                                                                   ØØØØØØ9Ø
C           + Ø.5*(H**2)*(X(I) + I*H + 1)**3 = Ø , I = 1,...,N      ØØØØ1ØØ
C                                                                   ØØØØ11Ø
C     WHERE H = 1/(N+1), AND X(Ø) = X(N+1) = Ø.                     ØØØØ12Ø
C                                                                   ØØØØ13Ø
C     **********                                                    ØØØØ14Ø
      INTEGER I,INT,INFO,LWA,N,NFCALL,NFEV,NWRITE                   ØØØØ15Ø
      DOUBLE PRECISION FNORM1,FNORM2,H,TEMP,TOL                     ØØØØ16Ø
      DOUBLE PRECISION X(1Ø),FVEC(1Ø),WA(13Ø)                       ØØØØ17Ø
      DOUBLE PRECISION DFLOAT                                       ØØØØ18Ø
      EXTERNAL BVP                                                  ØØØØ19Ø
      COMMON /REFNUM/ NFCALL                                        ØØØØ2ØØ
C                                                                   ØØØØ21Ø
C     LOGICAL OUTPUT UNIT IS ASSUMED TO BE NUMBER 6.                ØØØØ22Ø
C                                                                   ØØØØ23Ø
      DATA NWRITE /6/                                               ØØØØ24Ø
C                                                                   ØØØØ25Ø
      DFLOAT(INT) = INT                                             ØØØØ26Ø
      LWA = 13Ø                                                     ØØØØ27Ø
      TOL = 1.D-1Ø                                                  ØØØØ28Ø
      N = 1Ø                                                        ØØØØ29Ø
C                                                                   ØØØØ3ØØ
C     STARTING VALUES.                                              ØØØØ31Ø
C                                                                   ØØØØ32Ø
      H = 1.DØ/DFLOAT(N+1)                                          ØØØØ33Ø
```

```
      DO 1Ø I = 1, N                                                    ØØØØØ34Ø
         TEMP = DFLOAT(I)*H                                             ØØØØØ35Ø
         X(I) = TEMP*(TEMP - 1.DØ)                                      ØØØØØ36Ø
   1Ø    CONTINUE                                                       ØØØØØ37Ø
C                                                                       ØØØØØ38Ø
C     INITIAL MAX-NORM OF THE RESIDUALS.                                ØØØØØ39Ø
C                                                                       ØØØØØ4ØØ
      FNORM1 = Ø.DØ                                                     ØØØØØ41Ø
      DO 2Ø I = 1, N                                                    ØØØØØ42Ø
         CALL BVP(N,X,FVEC,I)                                           ØØØØØ43Ø
         FNORM1 = DMAX1(FNORM1,DABS(FVEC(I)))                           ØØØØØ44Ø
   2Ø    CONTINUE                                                       ØØØØØ45Ø
C                                                                       ØØØØØ46Ø
      NFCALL = Ø                                                        ØØØØØ47Ø
      CALL BRENT1(BVP,N,X,FVEC,TOL,INFO,WA,LWA)                         ØØØØØ48Ø
      NFEV = NFCALL/N                                                   ØØØØØ49Ø
C                                                                       ØØØØØ5ØØ
C     FINAL MAX-NORM OF THE RESIDUALS.                                  ØØØØØ51Ø
C                                                                       ØØØØØ52Ø
      FNORM2 = Ø.DØ                                                     ØØØØØ53Ø
      DO 3Ø I = 1, N                                                    ØØØØØ54Ø
         CALL BVP(N,X,FVEC,I)                                           ØØØØØ55Ø
         FNORM2 = DMAX1(FNORM2,DABS(FVEC(I)))                           ØØØØØ56Ø
   3Ø    CONTINUE                                                       ØØØØØ57Ø
C                                                                       ØØØØØ58Ø
      WRITE (NWRITE,1ØØØ) N,FNORM1,FNORM2,NFEV,INFO,(X(I),I=1,N)        ØØØØØ59Ø
      STOP                                                              ØØØØØ6ØØ
 1ØØØ FORMAT (5X,1ØH DIMENSION,I5,5X //                                 ØØØØØ61Ø
     1        5X,34H INITIAL MAX-NORM OF THE RESIDUALS,D15.7 //         ØØØØØ62Ø
     2        5X,34H FINAL MAX-NORM OF THE RESIDUALS  ,D15.7 //         ØØØØØ63Ø
     3        5X,33H NUMBER OF FUNCTION EVALUATIONS  ,I1Ø //            ØØØØØ64Ø
     4        5X,15H EXIT PARAMETER ,18X,I1Ø //                         ØØØØØ65Ø
     5        5X,27H FINAL APPROXIMATE SOLUTION  // (5X,5D15.7))      ØØØØØ66Ø
C                                                                       ØØØØØ67Ø
C     LAST CARD OF SAMPLE PROGRAM.                                      ØØØØØ68Ø
C                                                                       ØØØØØ69Ø
      END                                                               ØØØØØ7ØØ
      SUBROUTINE BVP(N,X,FVEC,IFLAG)                                    ØØØØØ71Ø
      INTEGER N,IFLAG                                                   ØØØØØ72Ø
      DOUBLE PRECISION X(N),FVEC(N)                                     ØØØØØ73Ø
C     **********                                                        ØØØØØ74Ø
C                                                                       ØØØØØ75Ø
C     SUBROUTINE BVP DEFINES THE BOUNDARY VALUE PROBLEM.                ØØØØØ76Ø
C                                                                       ØØØØØ77Ø
C     **********                                                        ØØØØØ78Ø
      INTEGER INT,NFCALL                                                ØØØØØ79Ø
      DOUBLE PRECISION H,TEMP,TEMP1,TEMP2                               ØØØØØ8ØØ
      DOUBLE PRECISION DFLOAT                                           ØØØØØ81Ø
      COMMON /REFNUM/ NFCALL                                            ØØØØØ82Ø
      DFLOAT(INT) = INT                                                 ØØØØØ83Ø
      H = 1.DØ/DFLOAT(N+1)                                              ØØØØØ84Ø
      TEMP = Ø.5DØ*((X(IFLAG) + DFLOAT(IFLAG)*H + 1.DØ)**3)             ØØØØØ85Ø
      TEMP1 = Ø.DØ                                                      ØØØØØ86Ø
      IF (IFLAG .NE. 1) TEMP1 = X(IFLAG-1)                              ØØØØØ87Ø
      TEMP2 = Ø.DØ                                                      ØØØØØ88Ø
      IF (IFLAG .NE. N) TEMP2 = X(IFLAG+1)                              ØØØØØ89Ø
      FVEC(IFLAG) = 2.DØ*X(IFLAG) - TEMP1 - TEMP2 + TEMP*H**2           ØØØØØ9ØØ
      NFCALL = NFCALL + 1                                               ØØØØØ91Ø
      RETURN                                                            ØØØØØ92Ø
C                                                                       ØØØØØ93Ø
C     LAST CARD OF SUBROUTINE BVP.                                      ØØØØØ94Ø
C                                                                       ØØØØØ95Ø
      END                                                               ØØØØØ96Ø
      SUBROUTINE BRENT1(FCN,N,X,FVEC,TOL,INFO,WA,LWA)                   ØØØØØØ1Ø
      INTEGER N,INFO,LWA                                                ØØØØØØ2Ø
      DOUBLE PRECISION TOL                                              ØØØØØØ3Ø
      DOUBLE PRECISION X(N),FVEC(N),WA(LWA)                             ØØØØØØ4Ø
      EXTERNAL FCN                                                      ØØØØØØ5Ø
C     **********                                                        ØØØØØØ6Ø
C                                                                       ØØØØØØ7Ø
C     SUBROUTINE BRENT1                                                 ØØØØØØ8Ø
C                                                                       ØØØØØØ9Ø
C     THE PURPOSE OF THIS SUBROUTINE IS TO FIND A ZERO OF               ØØØØ1ØØ
C     A SYSTEM OF N NONLINEAR EQUATIONS IN N VARIABLES BY A             ØØØØ11Ø
C     METHOD DUE TO R. BRENT. THIS IS DONE BY USING THE                 ØØØØ12Ø
C     MORE GENERAL NONLINEAR EQUATION SOLVER BRENTM.                    ØØØØ13Ø
```

```
C                                                                       ØØØØØ14Ø
C     THE SUBROUTINE STATEMENT IS                                       ØØØØØ15Ø
C                                                                       ØØØØØ16Ø
C       SUBROUTINE BRENT1(FCN,N,X,FVEC,TOL,INFO,WA,LWA)                 ØØØØØ17Ø
C                                                                       ØØØØØ18Ø
C     WHERE                                                             ØØØØØ19Ø
C                                                                       ØØØØØ2ØØ
C       FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH           ØØØØØ21Ø
C         CALCULATES COMPONENTS OF THE FUNCTION. FCN SHOULD BE          ØØØØØ22Ø
C         DECLARED IN AN EXTERNAL STATEMENT IN THE USER CALLING         ØØØØØ23Ø
C         PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.                    ØØØØØ24Ø
C                                                                       ØØØØØ25Ø
C         SUBROUTINE FCN(N,X,FVEC,IFLAG)                                ØØØØØ26Ø
C         INTEGER N,IFLAG                                               ØØØØØ27Ø
C         DOUBLE PRECISION X(N),FVEC(N)                                 ØØØØØ28Ø
C         ----------                                                    ØØØØØ29Ø
C         CALCULATE THE IFLAG-TH COMPONENT OF THE FUNCTION              ØØØØØ3ØØ
C         AND RETURN THIS VALUE IN FVEC(IFLAG).                         ØØØØØ31Ø
C         ----------                                                    ØØØØØ32Ø
C         RETURN                                                        ØØØØØ33Ø
C         END                                                           ØØØØØ34Ø
C                                                                       ØØØØØ35Ø
C         THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS        ØØØØØ36Ø
C         THE USER WANTS TO TERMINATE EXECUTION OF BRENT1.              ØØØØØ37Ø
C         IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.                 ØØØØØ38Ø
C                                                                       ØØØØØ39Ø
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER        ØØØØØ4ØØ
C         OF EQUATIONS AND VARIABLES.                                   ØØØØØ41Ø
C                                                                       ØØØØØ42Ø
C       X IS AN ARRAY OF LENGTH N. ON INPUT IT MUST CONTAIN             ØØØØØ43Ø
C         AN INITIAL ESTIMATE OF THE SOLUTION VECTOR. ON OUTPUT X       ØØØØØ44Ø
C         CONTAINS THE FINAL ESTIMATE OF THE SOLUTION VECTOR.           ØØØØØ45Ø
C                                                                       ØØØØØ46Ø
C       FVEC IS AN ARRAY OF LENGTH N. ON OUTPUT IT CONTAINS             ØØØØØ47Ø
C         THE FINAL RESIDUALS.                                          ØØØØØ48Ø
C                                                                       ØØØØØ49Ø
C       TOL IS A NONNEGATIVE INPUT VARIABLE. THE ALGORITHM CONVERGES    ØØØØØ5ØØ
C         IF EITHER ALL THE RESIDUALS ARE AT MOST TOL IN MAGNITUDE,     ØØØØØ51Ø
C         OR IF THE ALGORITHM ESTIMATES THAT THE RELATIVE ERROR         ØØØØØ52Ø
C         BETWEEN X AND THE SOLUTION IS AT MOST TOL.                    ØØØØØ53Ø
C                                                                       ØØØØØ54Ø
C       INFO IS AN INTEGER OUTPUT VARIABLE SET AS FOLLOWS. IF           ØØØØØ55Ø
C         THE USER HAS TERMINATED EXECUTION, INFO WILL BE SET TO        ØØØØØ56Ø
C         THE (NEGATIVE) VALUE OF IFLAG. SEE DESCRIPTION OF FCN.        ØØØØØ57Ø
C         OTHERWISE                                                     ØØØØØ58Ø
C                                                                       ØØØØØ59Ø
C           INFO = Ø   IMPROPER INPUT PARAMETERS.                       ØØØØØ6ØØ
C                                                                       ØØØØØ61Ø
C           INFO = 1   ALL RESIDUALS ARE AT MOST TOL IN MAGNITUDE.      ØØØØØ62Ø
C                                                                       ØØØØØ63Ø
C           INFO = 2   ALGORITHM ESTIMATES THAT THE RELATIVE ERROR      ØØØØØ64Ø
C                      BETWEEN X AND THE SOLUTION IS AT MOST TOL.       ØØØØØ65Ø
C                                                                       ØØØØØ66Ø
C           INFO = 3   CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD.  ØØØØØ67Ø
C                                                                       ØØØØØ68Ø
C           INFO = 4   NUMBER OF FUNCTION EVALUATIONS HAS REACHED OR    ØØØØØ69Ø
C                      EXCEEDED 5Ø*(N+3).                               ØØØØØ7ØØ
C                                                                       ØØØØØ71Ø
C           INFO = 5   APPROXIMATE JACOBIAN MATRIX IS SINGULAR.         ØØØØØ72Ø
C                                                                       ØØØØØ73Ø
C           INFO = 6   ITERATION IS NOT MAKING GOOD PROGRESS.           ØØØØØ74Ø
C                                                                       ØØØØØ75Ø
C           INFO = 7   ITERATION IS DIVERGING.                          ØØØØØ76Ø
C                                                                       ØØØØØ77Ø
C           INFO = 8   ITERATION IS CONVERGING, BUT TOL IS TOO          ØØØØØ78Ø
C                      SMALL, OR THE CONVERGENCE IS VERY SLOW           ØØØØØ79Ø
C                      DUE TO A JACOBIAN SINGULAR NEAR THE OUTPUT       ØØØØØ8ØØ
C                      X OR DUE TO BADLY SCALED VARIABLES.              ØØØØØ81Ø
C                                                                       ØØØØØ82Ø
C       WA IS A WORK ARRAY OF LENGTH LWA.                               ØØØØØ83Ø
C                                                                       ØØØØØ84Ø
C       LWA IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN          ØØØØØ85Ø
C         N*(N+3).                                                      ØØØØØ86Ø
C                                                                       ØØØØØ87Ø
C     SUBPROGRAMS REQUIRED                                              ØØØØØ88Ø
C                                                                       ØØØØØ89Ø
```

```
C       USER-SUPPLIED ..... FCN, BRENTM                                  ØØØØØ9ØØ
C                                                                        ØØØØØ91Ø
C       FORTRAN-SUPPLIED ... DLOG                                        ØØØØØ92Ø
C                                                                        ØØØØØ93Ø
C     **********                                                         ØØØØØ94Ø
      INTEGER I,IVAR,MAXFEV,MOPT,NFEV                                    ØØØØØ95Ø
      DOUBLE PRECISION EMAX,FTOL,TEMP,XTOL,ZERO                          ØØØØØ96Ø
      DOUBLE PRECISION DFLOAT                                            ØØØØØ97Ø
      DATA ZERO /Ø.DØ/                                                   ØØØØØ98Ø
      DFLOAT(IVAR) = IVAR                                                ØØØØØ99Ø
      INFO = Ø                                                           ØØØØ1ØØØ
C                                                                        ØØØØ1Ø1Ø
C     CHECK THE INPUT PARAMETERS FOR ERRORS.                             ØØØØ1Ø2Ø
C                                                                        ØØØØ1Ø3Ø
      IF (N .LE. Ø .OR. TOL .LT. ZERO .OR. LWA .LT. N*(N+3)) GO TO 3Ø    ØØØØ1Ø4Ø
C                                                                        ØØØØ1Ø5Ø
C     DETERMINE AN OPTIMAL VALUE FOR MOPT.                               ØØØØ1Ø6Ø
C                                                                        ØØØØ1Ø7Ø
      EMAX = ZERO                                                        ØØØØ1Ø8Ø
      DO 1Ø I = 1, N                                                     ØØØØ1Ø9Ø
         TEMP = DLOG(DFLOAT(I+1))/DFLOAT(N+2*I+1)                        ØØØØ11ØØ
         IF (TEMP .LT. EMAX) GO TO 2Ø                                    ØØØØ111Ø
         MOPT = I                                                        ØØØØ112Ø
         EMAX = TEMP                                                     ØØØØ113Ø
   1Ø    CONTINUE                                                        ØØØØ114Ø
   2Ø CONTINUE                                                          ØØØØ115Ø
C                                                                        ØØØØ116Ø
C     CALL BRENTM.                                                       ØØØØ117Ø
C                                                                        ØØØØ118Ø
      MAXFEV = 5Ø*(N + 3)                                                ØØØØ119Ø
      FTOL = TOL                                                         ØØØØ12ØØ
      XTOL = TOL                                                         ØØØØ121Ø
      CALL BRENTM(FCN,N,X,FVEC,FTOL,XTOL,MAXFEV,MOPT,                    ØØØØ122Ø
     1            INFO,NFEV,WA(3*N+1),N,WA(1),WA(N+1),WA(2*N+1))         ØØØØ123Ø
   3Ø CONTINUE                                                          ØØØØ124Ø
      RETURN                                                             ØØØØ125Ø
C                                                                        ØØØØ126Ø
C     LAST CARD OF SUBROUTINE BRENT1.                                    ØØØØ127Ø
C                                                                        ØØØØ128Ø
      END                                                                ØØØØ129Ø
      SUBROUTINE BRENTM(FCN,N,X,FVEC,FTOL,XTOL,MAXFEV,MOPT,              ØØØØØØ1Ø
     1                  INFO,NFEV,Q,LDQ,SIGMA,WA1,WA2)                   ØØØØØØ2Ø
      INTEGER N,MAXFEV,MOPT,INFO,NFEV,LDQ                                ØØØØØØ3Ø
      DOUBLE PRECISION FTOL,XTOL                                         ØØØØØØ4Ø
      DOUBLE PRECISION X(N),FVEC(N),Q(LDQ,N),SIGMA(N),WA1(N),WA2(N)      ØØØØØØ5Ø
C     **********                                                         ØØØØØØ6Ø
C                                                                        ØØØØØØ7Ø
C     SUBROUTINE BRENTM                                                  ØØØØØØ8Ø
C                                                                        ØØØØØØ9Ø
C     THE PURPOSE OF THIS SUBROUTINE IS TO FIND A ZERO TO                ØØØØØ1ØØ
C     A SYSTEM OF N NONLINEAR EQUATIONS IN N VARIABLES BY A              ØØØØØ11Ø
C     METHOD DUE TO R. BRENT.                                            ØØØØØ12Ø
C                                                                        ØØØØØ13Ø
C     THE SUBROUTINE STATEMENT IS                                        ØØØØØ14Ø
C                                                                        ØØØØØ15Ø
C       SUBROUTINE BRENTM(FCN,N,X,FVEC,FTOL,XTOL,MAXFEV,MOPT,            ØØØØØ16Ø
C                         INFO,NFEV,Q,LDQ,SIGMA,WA1,WA2)                 ØØØØØ17Ø
C                                                                        ØØØØØ18Ø
C     WHERE                                                              ØØØØØ19Ø
C                                                                        ØØØØØ2ØØ
C       FCN IS THE NAME OF THE USER-SUPPLIED SUBROUTINE WHICH            ØØØØØ21Ø
C         CALCULATES COMPONENTS OF THE FUNCTION. FCN SHOULD BE           ØØØØØ22Ø
C         DECLARED IN AN EXTERNAL STATEMENT IN THE USER CALLING          ØØØØØ23Ø
C         PROGRAM, AND SHOULD BE WRITTEN AS FOLLOWS.                     ØØØØØ24Ø
C                                                                        ØØØØØ25Ø
C         SUBROUTINE FCN(N,X,FVEC,IFLAG)                                 ØØØØØ26Ø
C         INTEGER N,IFLAG                                                ØØØØØ27Ø
C         DOUBLE PRECISION X(N),FVEC(N)                                  ØØØØØ28Ø
C         ----------                                                     ØØØØØ29Ø
C         CALCULATE THE IFLAG-TH COMPONENT OF THE FUNCTION               ØØØØØ3ØØ
C         AND RETURN THIS VALUE IN FVEC(IFLAG).                          ØØØØØ31Ø
C         ----------                                                     ØØØØØ32Ø
C         RETURN                                                         ØØØØØ33Ø
C         END                                                            ØØØØØ34Ø
C                                                                        ØØØØØ35Ø
C         THE VALUE OF IFLAG SHOULD NOT BE CHANGED BY FCN UNLESS         ØØØØØ36Ø
```

```
C         THE USER WANTS TO TERMINATE EXECUTION OF BRENTM.                  ØØØØØ37Ø
C         IN THIS CASE SET IFLAG TO A NEGATIVE INTEGER.                     ØØØØØ38Ø
C                                                                           ØØØØØ39Ø
C       N IS A POSITIVE INTEGER INPUT VARIABLE SET TO THE NUMBER OF         ØØØØØ4ØØ
C         EQUATIONS AND VARIABLES.                                          ØØØØØ41Ø
C                                                                           ØØØØØ42Ø
C       X IS AN ARRAY OF LENGTH N. ON INPUT IT MUST CONTAIN                 ØØØØØ43Ø
C         AN ESTIMATE TO THE SOLUTION OF THE SYSTEM OF EQUATIONS.           ØØØØØ44Ø
C         ON OUTPUT X CONTAINS THE FINAL ESTIMATE TO THE SOLUTION           ØØØØØ45Ø
C         OF THE SYSTEM OF EQUATIONS.                                       ØØØØØ46Ø
C                                                                           ØØØØØ47Ø
C       FVEC IS AN ARRAY OF LENGTH N. ON OUTPUT IT CONTAINS                 ØØØØØ48Ø
C         THE FINAL RESIDUALS.                                              ØØØØØ49Ø
C                                                                           ØØØØØ5ØØ
C       FTOL IS A NONNEGATIVE INPUT VARIABLE. CONVERGENCE                   ØØØØØ51Ø
C         OCCURS IF ALL RESIDUALS ARE AT MOST FTOL IN MAGNITUDE.            ØØØØØ52Ø
C                                                                           ØØØØØ53Ø
C       XTOL IS A NONNEGATIVE INPUT VARIABLE. CONVERGENCE                   ØØØØØ54Ø
C         OCCURS IF THE RELATIVE ERROR BETWEEN TWO SUCCESSIVE               ØØØØØ55Ø
C         ITERATES IS AT MOST XTOL.                                         ØØØØØ56Ø
C                                                                           ØØØØØ57Ø
C       MAXFEV IS A POSITIVE INTEGER INPUT VARIABLE. TERMINATION            ØØØØØ58Ø
C         OCCURS IF THE NUMBER OF FUNCTION EVALUATIONS IS AT                ØØØØØ59Ø
C         LEAST MAXFEV BY THE END OF AN ITERATION. IN BRENTM,               ØØØØØ6ØØ
C         A FUNCTION EVALUATION CORRESPONDS TO N CALLS TO FCN.              ØØØØØ61Ø
C                                                                           ØØØØØ62Ø
C       MOPT IS A POSITIVE INTEGER INPUT VARIABLE. MOPT SPECIFIES           ØØØØØ63Ø
C         THE NUMBER OF TIMES THAT THE APPROXIMATE JACOBIAN IS              ØØØØØ64Ø
C         USED DURING EACH ITERATION WHICH EMPLOYS ITERATIVE                ØØØØØ65Ø
C         REFINEMENT. IF MOPT IS 1, NO ITERATIVE REFINEMENT WILL            ØØØØØ66Ø
C         BE DONE. MAXIMUM EFFICIENCY IS USUALLY OBTAINED IF                ØØØØØ67Ø
C         MOPT MAXIMIZES LOG(K+1)/(N+2*K+1) FOR K = 1,...,N.                ØØØØØ68Ø
C                                                                           ØØØØØ69Ø
C       INFO IS AN INTEGER OUTPUT VARIABLE SET AS FOLLOWS. IF               ØØØØØ7ØØ
C         THE USER HAS TERMINATED EXECUTION, INFO WILL BE SET TO            ØØØØØ71Ø
C         THE (NEGATIVE) VALUE OF IFLAG. SEE DESCRIPTION OF FCN.            ØØØØØ72Ø
C         OTHERWISE                                                         ØØØØØ73Ø
C                                                                           ØØØØØ74Ø
C           INFO = Ø   IMPROPER INPUT PARAMETERS.                           ØØØØØ75Ø
C                                                                           ØØØØØ76Ø
C           INFO = 1   ALL RESIDUALS ARE AT MOST FTOL IN MAGNITUDE.         ØØØØØ77Ø
C                                                                           ØØØØØ78Ø
C           INFO = 2   RELATIVE ERROR BETWEEN TWO SUCCESSIVE ITERATES       ØØØØØ79Ø
C                      IS AT MOST XTOL.                                     ØØØØØ8ØØ
C                                                                           ØØØØØ81Ø
C           INFO = 3   CONDITIONS FOR INFO = 1 AND INFO = 2 BOTH HOLD.      ØØØØØ82Ø
C                                                                           ØØØØØ83Ø
C           INFO = 4   NUMBER OF FUNCTION EVALUATIONS HAS REACHED OR        ØØØØØ84Ø
C                      EXCEEDED MAXFEV.                                     ØØØØØ85Ø
C                                                                           ØØØØØ86Ø
C           INFO = 5   APPROXIMATE JACOBIAN MATRIX IS SINGULAR.             ØØØØØ87Ø
C                                                                           ØØØØØ88Ø
C           INFO = 6   ITERATION IS NOT MAKING GOOD PROGRESS.               ØØØØØ89Ø
C                                                                           ØØØØØ9ØØ
C           INFO = 7   ITERATION IS DIVERGING.                              ØØØØØ91Ø
C                                                                           ØØØØØ92Ø
C           INFO = 8   ITERATION IS CONVERGING, BUT XTOL IS TOO             ØØØØØ93Ø
C                      SMALL, OR THE CONVERGENCE IS VERY SLOW               ØØØØØ94Ø
C                      DUE TO A JACOBIAN SINGULAR NEAR THE OUTPUT           ØØØØØ95Ø
C                      X OR DUE TO BADLY SCALED VARIABLES.                  ØØØØØ96Ø
C                                                                           ØØØØØ97Ø
C       NFEV IS AN INTEGER OUTPUT VARIABLE SET TO THE NUMBER OF             ØØØØØ98Ø
C         FUNCTION EVALUATIONS USED IN PRODUCING X. IN BRENTM,              ØØØØØ99Ø
C         A FUNCTION EVALUATION CORRESPONDS TO N CALLS TO FCN.              ØØØØ1ØØØ
C                                                                           ØØØØ1Ø1Ø
C       Q IS AN N BY N ARRAY. IF JAC DENOTES THE APPROXIMATE                ØØØØ1Ø2Ø
C         JACOBIAN, THEN ON OUTPUT Q IS (A MULTIPLE OF) AN                  ØØØØ1Ø3Ø
C         ORTHOGONAL MATRIX SUCH THAT JAC*Q IS A LOWER TRIANGULAR           ØØØØ1Ø4Ø
C         MATRIX. ONLY THE DIAGONAL ELEMENTS OF JAC*Q NEED                  ØØØØ1Ø5Ø
C         TO BE STORED, AND THESE CAN BE FOUND IN SIGMA.                    ØØØØ1Ø6Ø
C                                                                           ØØØØ1Ø7Ø
C       LDQ IS A POSITIVE INTEGER INPUT VARIABLE NOT LESS THAN N            ØØØØ1Ø8Ø
C         WHICH SPECIFIES THE LEADING DIMENSION OF THE ARRAY Q.             ØØØØ1Ø9Ø
C                                                                           ØØØØ11ØØ
C       SIGMA IS A LINEAR ARRAY OF LENGTH N. ON OUTPUT SIGMA                ØØØØ111Ø
C         CONTAINS THE DIAGONAL ELEMENTS OF THE MATRIX JAC*Q.               ØØØØ112Ø
```

```
C         SEE DESCRIPTION OF Q.                                          ØØØØ113Ø
C                                                                        ØØØØ114Ø
C       WA1 AND WA2 ARE LINEAR WORK ARRAYS OF LENGTH N.                  ØØØØ115Ø
C                                                                        ØØØØ116Ø
C     SUBPROGRAMS REQUIRED                                               ØØØØ117Ø
C                                                                        ØØØØ118Ø
C       USER-SUPPLIED ..... FCN                                          ØØØØ119Ø
C                                                                        ØØØØ12ØØ
C       FORTRAN-SUPPLIED ... DABS,DMAX1,DSQRT,DSIGN                      ØØØØ121Ø
C                                                                        ØØØØ122Ø
C     **********                                                         ØØØØ123Ø
      INTEGER I,IFLAG,J,K,M,NFCALL,NIER6,NIER7,NIER8,NSING               ØØØØ124Ø
      LOGICAL CONV                                                       ØØØØ125Ø
      DOUBLE PRECISION DELTA,DIFIT,DIFIT1,EPS,EPSMCH,ETA,FKY,FKZ,        ØØØØ126Ø
     1       FNORM,FNORM1,H,PØ5,SCALE,SKNORM,TEMP,XNORM,ZERO             ØØØØ127Ø
      DATA ZERO,PØ5,SCALE /Ø.DØ,5.D-2,1.D1/                              ØØØØ128Ø
C                                                                        ØØØØ129Ø
C     WARNING.                                                           ØØØØ13ØØ
C                                                                        ØØØØ131Ø
C     THIS IS AN IBM CODE. TO RUN THIS CODE ON OTHER MACHINES IT         ØØØØ132Ø
C     IS NECESSARY TO CHANGE THE VALUE OF THE MACHINE PRECISION          ØØØØ133Ø
C     EPSMCH. THE MACHINE PRECISION IS THE SMALLEST FLOATING             ØØØØ134Ø
C     POINT NUMBER EPSMCH SUCH THAT                                      ØØØØ135Ø
C                                                                        ØØØØ136Ø
C           1 + EPSMCH .GT. 1                                            ØØØØ137Ø
C                                                                        ØØØØ138Ø
C     IN WORKING PRECISION. IF IN DOUBT ABOUT THE VALUE OF               ØØØØ139Ø
C     EPSMCH, THEN THE FOLLOWING PROGRAM SEGMENT DETERMINES              ØØØØ14ØØ
C     EPSMCH ON MOST MACHINES.                                           ØØØØ141Ø
C                                                                        ØØØØ142Ø
C     EPSMCH = Ø.5DØ                                                     ØØØØ143Ø
C   1 CONTINUE                                                           ØØØØ144Ø
C     IF (1.DØ+EPSMCH .EQ. 1.DØ) GO TO 2                                 ØØØØ145Ø
C     EPSMCH = Ø.5DØ*EPSMCH                                              ØØØØ146Ø
C     GO TO 1                                                            ØØØØ147Ø
C   2 CONTINUE                                                           ØØØØ148Ø
C     EPSMCH = 2.DØ*EPSMCH                                               ØØØØ149Ø
C                                                                        ØØØØ15ØØ
C     THE IBM DOUBLE PRECISION EPSMCH.                                   ØØØØ151Ø
C                                                                        ØØØØ152Ø
      EPSMCH = 16.DØ**(-13)                                              ØØØØ153Ø
C                                                                        ØØØØ154Ø
      INFO = Ø                                                           ØØØØ155Ø
      IFLAG = Ø                                                          ØØØØ156Ø
      NFEV = Ø                                                           ØØØØ157Ø
      NFCALL = Ø                                                         ØØØØ158Ø
C                                                                        ØØØØ159Ø
C     CHECK THE INPUT PARAMETERS FOR ERRORS.                             ØØØØ16ØØ
C                                                                        ØØØØ161Ø
      IF (N .LE. Ø .OR. FTOL .LT. ZERO .OR. XTOL .LT. ZERO .OR.          ØØØØ162Ø
     1    MAXFEV .LE. Ø .OR. MOPT .LE. Ø .OR. LDQ .LT. N) GO TO 22Ø      ØØØØ163Ø
C                                                                        ØØØØ164Ø
C     INITIALIZE SOME OF THE VARIABLES.                                  ØØØØ165Ø
C                                                                        ØØØØ166Ø
      NIER6 = -1                                                         ØØØØ167Ø
      NIER7 = -1                                                         ØØØØ168Ø
      NIER8 = Ø                                                          ØØØØ169Ø
      FNORM = ZERO                                                       ØØØØ17ØØ
      DIFIT = ZERO                                                       ØØØØ171Ø
      XNORM = ZERO                                                       ØØØØ172Ø
      DO 1Ø I = 1, N                                                     ØØØØ173Ø
         XNORM = DMAX1(XNORM,DABS(X(I)))                                 ØØØØ174Ø
   1Ø    CONTINUE                                                       ØØØØ175Ø
      EPS = DSQRT(EPSMCH)                                                ØØØØ176Ø
      DELTA = SCALE*XNORM                                                ØØØØ177Ø
      IF (XNORM .EQ. ZERO) DELTA = SCALE                                 ØØØØ178Ø
C                                                                        ØØØØ179Ø
C     ENTER THE PRINCIPAL ITERATION.                                     ØØØØ18ØØ
C                                                                        ØØØØ181Ø
   2Ø CONTINUE                                                           ØØØØ182Ø
C                                                                        ØØØØ183Ø
C     TO PRINT THE ITERATES, PLACE WRITE STATEMENTS                      ØØØØ184Ø
C     FOR THE VECTOR X HERE.                                             ØØØØ185Ø
C                                                                        ØØØØ186Ø
      NSING = N                                                          ØØØØ187Ø
      FNORM1 = FNORM                                                     ØØØØ188Ø
```

```
      DIFIT1 = DIFIT                                                    ØØØØ189Ø
      FNORM = ZERO                                                      ØØØØ19ØØ
C                                                                       ØØØØ191Ø
C     COMPUTE THE STEP H FOR THE DIVIDED DIFFERENCE WHICH               ØØØØ192Ø
C     APPROXIMATES THE K-TH ROW OF THE JACOBIAN MATRIX.                 ØØØØ193Ø
C                                                                       ØØØØ194Ø
      H = EPS*XNORM                                                     ØØØØ195Ø
      IF (H .EQ. ZERO) H = EPS                                          ØØØØ196Ø
      DO 4Ø J = 1, N                                                    ØØØØ197Ø
         DO 3Ø I = 1, N                                                 ØØØØ198Ø
            Q(I,J) = ZERO                                               ØØØØ199Ø
   3Ø       CONTINUE                                                   ØØØØ2ØØØ
         Q(J,J) = H                                                     ØØØØ2Ø1Ø
         WA1(J) = X(J)                                                  ØØØØ2Ø2Ø
   4Ø    CONTINUE                                                      ØØØØ2Ø3Ø
C                                                                       ØØØØ2Ø4Ø
C     ENTER A SUBITERATION.                                             ØØØØ2Ø5Ø
C                                                                       ØØØØ2Ø6Ø
      DO 15Ø K = 1, N                                                   ØØØØ2Ø7Ø
         IFLAG = K                                                      ØØØØ2Ø8Ø
         CALL FCN(N,WA1,FVEC,IFLAG)                                     ØØØØ2Ø9Ø
         FKY = FVEC(K)                                                  ØØØØ21ØØ
         NFCALL = NFCALL + 1                                            ØØØØ211Ø
         NFEV = NFCALL/N                                                ØØØØ212Ø
         IF (IFLAG .LT. Ø) GO TO 23Ø                                    ØØØØ213Ø
         FNORM = DMAX1(FNORM,DABS(FKY))                                 ØØØØ214Ø
C                                                                       ØØØØ215Ø
C        COMPUTE THE K-TH ROW OF THE JACOBIAN MATRIX.                   ØØØØ216Ø
C                                                                       ØØØØ217Ø
         DO 6Ø J = K, N                                                 ØØØØ218Ø
            DO 5Ø I = 1, N                                              ØØØØ219Ø
               WA2(I) = WA1(I) + Q(I,J)                                 ØØØØ22ØØ
   5Ø          CONTINUE                                                ØØØØ221Ø
            CALL FCN(N,WA2,FVEC,IFLAG)                                  ØØØØ222Ø
            FKZ = FVEC(K)                                               ØØØØ223Ø
            NFCALL = NFCALL + 1                                         ØØØØ224Ø
            NFEV = NFCALL/N                                             ØØØØ225Ø
            IF (IFLAG .LT. Ø) GO TO 23Ø                                 ØØØØ226Ø
            SIGMA(J) = FKZ - FKY                                        ØØØØ227Ø
   6Ø       CONTINUE                                                   ØØØØ228Ø
         FVEC(K) = FKY                                                  ØØØØ229Ø
C                                                                       ØØØØ23ØØ
C        COMPUTE THE HOUSEHOLDER TRANSFORMATION TO REDUCE THE K-TH ROW  ØØØØ231Ø
C        OF THE JACOBIAN MATRIX TO A MULTIPLE OF THE K-TH UNIT VECTOR.  ØØØØ232Ø
C                                                                       ØØØØ233Ø
         ETA = ZERO                                                     ØØØØ234Ø
         DO 7Ø I = K, N                                                 ØØØØ235Ø
            ETA = DMAX1(ETA,DABS(SIGMA(I)))                             ØØØØ236Ø
   7Ø       CONTINUE                                                   ØØØØ237Ø
         IF (ETA .EQ. ZERO) GO TO 15Ø                                   ØØØØ238Ø
         NSING = NSING - 1                                              ØØØØ239Ø
         SKNORM = ZERO                                                  ØØØØ24ØØ
         DO 8Ø I = K, N                                                 ØØØØ241Ø
            SIGMA(I) = SIGMA(I)/ETA                                     ØØØØ242Ø
            SKNORM = SKNORM + SIGMA(I)**2                               ØØØØ243Ø
   8Ø       CONTINUE                                                   ØØØØ244Ø
         SKNORM = DSQRT(SKNORM)                                         ØØØØ245Ø
         IF (SIGMA(K) .LT. ZERO) SKNORM = -SKNORM                       ØØØØ246Ø
         SIGMA(K) = SIGMA(K) + SKNORM                                   ØØØØ247Ø
C                                                                       ØØØØ248Ø
C        APPLY THE TRANSFORMATION AND COMPUTE THE MATRIX Q.             ØØØØ249Ø
C                                                                       ØØØØ25ØØ
         DO 9Ø I = 1, N                                                 ØØØØ251Ø
            WA2(I) = ZERO                                               ØØØØ252Ø
   9Ø       CONTINUE                                                   ØØØØ253Ø
         DO 11Ø J = K, N                                                ØØØØ254Ø
            TEMP = SIGMA(J)                                             ØØØØ255Ø
            DO 1ØØ I = 1, N                                             ØØØØ256Ø
               WA2(I) = WA2(I) + TEMP*Q(I,J)                            ØØØØ257Ø
  1ØØ          CONTINUE                                                ØØØØ258Ø
  11Ø       CONTINUE                                                   ØØØØ259Ø
         DO 13Ø J = K, N                                                ØØØØ26ØØ
            TEMP = SIGMA(J)/(SKNORM*SIGMA(K))                           ØØØØ261Ø
            DO 12Ø I = 1, N                                             ØØØØ262Ø
               Q(I,J) = Q(I,J) - TEMP*WA2(I)                            ØØØØ263Ø
  12Ø          CONTINUE                                                ØØØØ264Ø
```

```
  13Ø       CONTINUE                                                    ØØØØ265Ø
C                                                                       ØØØØ266Ø
C        COMPUTE THE SUBITERATE.                                        ØØØØ267Ø
C                                                                       ØØØØ268Ø
         SIGMA(K) = SKNORM*ETA                                          ØØØØ269Ø
         TEMP = FKY/SIGMA(K)                                            ØØØØ27ØØ
         IF (H*DABS(TEMP) .GT. DELTA) TEMP = DSIGN(DELTA/H,TEMP)        ØØØØ271Ø
         DO 14Ø I = 1, N                                                ØØØØ272Ø
            WA1(I) = WA1(I) + TEMP*Q(I,K)                               ØØØØ273Ø
  14Ø       CONTINUE                                                    ØØØØ274Ø
  15Ø    CONTINUE                                                       ØØØØ275Ø
C                                                                       ØØØØ276Ø
C     COMPUTE THE NORMS OF THE ITERATE AND CORRECTION VECTOR.           ØØØØ277Ø
C                                                                       ØØØØ278Ø
      XNORM = ZERO                                                      ØØØØ279Ø
      DIFIT = ZERO                                                      ØØØØ28ØØ
      DO 16Ø I = 1, N                                                   ØØØØ281Ø
         XNORM = DMAX1(XNORM,DABS(WA1(I)))                              ØØØØ282Ø
         DIFIT = DMAX1(DIFIT,DABS(X(I)-WA1(I)))                         ØØØØ283Ø
         X(I) = WA1(I)                                                  ØØØØ284Ø
  16Ø    CONTINUE                                                       ØØØØ285Ø
C                                                                       ØØØØ286Ø
C     UPDATE THE BOUND ON THE CORRECTION VECTOR.                        ØØØØ287Ø
C                                                                       ØØØØ288Ø
      DELTA = DMAX1(DELTA,SCALE*XNORM)                                  ØØØØ289Ø
C                                                                       ØØØØ29ØØ
C     DETERMINE THE PROGRESS OF THE ITERATION.                          ØØØØ291Ø
C                                                                       ØØØØ292Ø
      CONV = (FNORM .LT. FNORM1 .AND. DIFIT .LT. DIFIT1 .AND.           ØØØØ293Ø
     1        NSING .EQ. Ø)                                             ØØØØ294Ø
      NIER6 = NIER6 + 1                                                 ØØØØ295Ø
      NIER7 = NIER7 + 1                                                 ØØØØ296Ø
      NIER8 = NIER8 + 1                                                 ØØØØ297Ø
      IF (CONV) NIER6 = Ø                                               ØØØØ298Ø
      IF (FNORM .LT. FNORM1 .OR. DIFIT .LT. DIFIT1) NIER7 = Ø           ØØØØ299Ø
      IF (DIFIT .GT. EPS*XNORM) NIER8 = Ø                               ØØØØ3ØØØ
C                                                                       ØØØØ3Ø1Ø
C     TESTS FOR CONVERGENCE.                                            ØØØØ3Ø2Ø
C                                                                       ØØØØ3Ø3Ø
      IF (FNORM .LE. FTOL) INFO = 1                                     ØØØØ3Ø4Ø
      IF (DIFIT .LE. XTOL*XNORM .AND. CONV) INFO = 2                    ØØØØ3Ø5Ø
      IF (FNORM .LE. FTOL .AND. INFO .EQ. 2) INFO = 3                   ØØØØ3Ø6Ø
      IF (INFO .NE. Ø) GO TO 23Ø                                        ØØØØ3Ø7Ø
C                                                                       ØØØØ3Ø8Ø
C     TESTS FOR TERMINATION.                                            ØØØØ3Ø9Ø
C                                                                       ØØØØ31ØØ
      IF (NFEV .GE. MAXFEV) INFO = 4                                    ØØØØ311Ø
      IF (NSING .EQ. N) INFO = 5                                        ØØØØ312Ø
      IF (NIER6 .EQ. 5) INFO = 6                                        ØØØØ313Ø
      IF (NIER7 .EQ. 3) INFO = 7                                        ØØØØ314Ø
      IF (NIER8 .EQ. 4) INFO = 8                                        ØØØØ315Ø
      IF (INFO .NE. Ø) GO TO 23Ø                                        ØØØØ316Ø
C                                                                       ØØØØ317Ø
C     ITERATIVE REFINEMENT IS USED IF THE ITERATION IS CONVERGING.      ØØØØ318Ø
C                                                                       ØØØØ319Ø
      IF (.NOT. CONV .OR. DIFIT .GT. PØ5*XNORM .OR. MOPT .EQ. 1)        ØØØØ32ØØ
     1    GO TO 22Ø                                                     ØØØØ321Ø
C                                                                       ØØØØ322Ø
C     START ITERATIVE REFINEMENT.                                       ØØØØ323Ø
C                                                                       ØØØØ324Ø
      DO 21Ø M = 2, MOPT                                                ØØØØ325Ø
         FNORM1 = FNORM                                                 ØØØØ326Ø
         FNORM = ZERO                                                   ØØØØ327Ø
         DO 19Ø K = 1, N                                                ØØØØ328Ø
            IFLAG = K                                                   ØØØØ329Ø
            CALL FCN(N,WA1,FVEC,IFLAG)                                  ØØØØ33ØØ
            FKY = FVEC(K)                                               ØØØØ331Ø
            NFCALL = NFCALL + 1                                         ØØØØ332Ø
            NFEV = NFCALL/N                                             ØØØØ333Ø
            IF (IFLAG .LT. Ø) GO TO 23Ø                                 ØØØØ334Ø
            FNORM = DMAX1(FNORM,DABS(FKY))                              ØØØØ335Ø
C                                                                       ØØØØ336Ø
C           ITERATIVE REFINEMENT IS TERMINATED IF IT DOES NOT           ØØØØ337Ø
C           GIVE A REDUCTION OF THE RESIDUALS.                          ØØØØ338Ø
C                                                                       ØØØØ339Ø
            IF (FNORM .LT. FNORM1) GO TO 17Ø                            ØØØØ34ØØ
```

```
            FNORM = FNORM1                                                  ØØØØ341Ø
            GO TO 22Ø                                                       ØØØØ342Ø
  17Ø       CONTINUE                                                        ØØØØ343Ø
            TEMP = FKY/SIGMA(K)                                             ØØØØ344Ø
            DO 18Ø I = 1, N                                                 ØØØØ345Ø
               WA1(I) = WA1(I) + TEMP*Q(I,K)                                ØØØØ346Ø
  18Ø       CONTINUE                                                        ØØØØ347Ø
  19Ø       CONTINUE                                                        ØØØØ348Ø
C                                                                           ØØØØ349Ø
C        COMPUTE THE NORMS OF THE ITERATE AND CORRECTION VECTOR.            ØØØØ35ØØ
C                                                                           ØØØØ351Ø
         XNORM = ZERO                                                       ØØØØ352Ø
         DIFIT = ZERO                                                       ØØØØ353Ø
         DO 2ØØ I = 1, N                                                    ØØØØ354Ø
            XNORM = DMAX1(XNORM,DABS(WA1(I)))                               ØØØØ355Ø
            DIFIT = DMAX1(DIFIT,DABS(X(I)-WA1(I)))                          ØØØØ356Ø
            X(I) = WA1(I)                                                   ØØØØ357Ø
  2ØØ       CONTINUE                                                        ØØØØ358Ø
C                                                                           ØØØØ359Ø
C        STOPPING CRITERIA FOR ITERATIVE REFINEMENT.                        ØØØØ36ØØ
C                                                                           ØØØØ361Ø
         IF (FNORM .LE. FTOL) INFO = 1                                      ØØØØ362Ø
         IF (DIFIT .LE. XTOL*XNORM) INFO = 2                                ØØØØ363Ø
         IF (FNORM .LE. FTOL .AND. INFO .EQ. 2) INFO = 3                    ØØØØ364Ø
         IF (NFEV .GE. MAXFEV .AND. INFO .EQ. Ø) INFO = 4                   ØØØØ365Ø
         IF (INFO .NE. Ø) GO TO 23Ø                                         ØØØØ366Ø
  21Ø    CONTINUE                                                          ØØØØ367Ø
  22Ø CONTINUE                                                            ØØØØ368Ø
C                                                                           ØØØØ369Ø
C     END OF THE ITERATIVE REFINEMENT.                                      ØØØØ37ØØ
C                                                                           ØØØØ371Ø
      GO TO 2Ø                                                            ØØØØ372Ø
C                                                                           ØØØØ373Ø
C     TERMINATION, EITHER NORMAL OR USER IMPOSED.                           ØØØØ374Ø
C                                                                           ØØØØ375Ø
  23Ø CONTINUE                                                            ØØØØ376Ø
      IF (IFLAG .LT. Ø) INFO = IFLAG                                        ØØØØ377Ø
      RETURN                                                                ØØØØ378Ø
C                                                                           ØØØØ379Ø
C     LAST CARD OF SUBROUTINE BRENTM.                                       ØØØØ38ØØ
C                                                                           ØØØØ381Ø
      END                                                                   ØØØØ382Ø
```

ALGORITHM 555
Chow–Yorke Algorithm for Fixed Points or Zeros of C^2 Maps [C5]

LAYNE T. WATSON
Michigan State University
and
DAN FENNER
Johns Hopkins Applied Physics Laboratory

Key Words and Phrases: fixed point, zero, nonlinear system, homotopy method, continuation method, parameterized nonlinear system, zero curve of a homotopy map, fixed points of nonlinear systems, zeros of nonlinear systems
CR Categories: 5.15
Language: Fortran

DESCRIPTION

Introduction

Homotopy methods (also known as continuation methods or methods of incremental loading) for computing zeros of nonlinear systems are well known [1, 4–11, 13]. In abstract terms a homotopy method is as follows: Let B be a Banach space and $f: B \to B$ the function whose zero is desired. Let $s: B \to B$ be a simple function with a known zero. Construct a continuous map (the homotopy) $\Phi: B \times [0, 1] \to B$ such that $\Phi(x, 0) = s(x)$ and $\Phi(x, 1) = f(x)$. Then by solving the equation $\Phi(x, \lambda) = 0$ in $B \times [0, 1]$, one attempts to move from the known zero of $s(x)$ (at $\lambda = 0$) to the unknown zero of $f(x)$ (at $\lambda = 1$). There is a considerable amount of theory concerning when this procedure will work [11], but, in general, moving from a zero of $s(x)$ to a zero of $f(x)$ may or may not be possible. Sometimes λ is treated as an independent variable [4, 11], and sometimes λ is a dependent variable with arc length or some other parameter as the independent variable [7, 9, 10, 13].

For the calculation of (Brouwer) fixed points, the supporting theory is much more satisfactory. Let B be the closed unit ball (or any compact, convex subset) in E^n, and let $f: B \to B$ be a C^2 map. Chow et al. [3] have proved the following powerful theorem: Define $\rho_a: [0, 1) \times B \to E^n$ by

$$\rho_a(\lambda, x) = \lambda(x - f(x)) + (1 - \lambda)(x - a)$$

where $f: B \to B$ is a C^2 map such that the Jacobian matrix of $x - f(x)$ is nonsingular at every fixed point of f. Then for almost all a in the interior of B, the

Received 21 June 1977; revised 21 June 1978; accepted 30 July 1979

Authors' present addresses: L.T. Watson, Department of Computer Science, Virginia Polytechnic Institute, Blacksburg, VA. 24061; D. Fenner, Johns Hopkins Applied Physics Laboratory, Johns Hopkins Road, Laurel, MD 20810.

set $\{(\lambda, x) \mid 0 \le \lambda < 1, x \in B, \rho_a(\lambda, x) = 0\}$ of zeros of ρ_a consists of

(1) a finite number of closed loops (having finite length) in $(0, 1) \times B$;
(2) a finite number of arcs (having finite length) in $(0, 1) \times B$ with endpoints in $\{1\} \times B$;
(3) a curve of finite length starting at $(0, a)$ and ending at $(1, \bar{x})$, where $\bar{x} \in B$ is a fixed point of f.

The curves in (1), (2), and (3) are disjoint and continuously differentiable.

In other words, with probability 1 there is a zero curve of $\rho_a(\lambda, x)$ emanating from $(0, a)$ which reaches a fixed point $\bar{x}$ of f (at $\lambda = 1$). This zero curve is smooth and has finite length. Thus computing a fixed point of f merely amounts to tracking a (smooth) zero curve of $\rho_a(\lambda, x)$. Note that the asumptions on $f(C^2$ smoothness and nonsingularity of the Jacobian matrix of $x - f(x)$ at the fixed point) are quite mild considering that global convergence is guaranteed with probability 1.

A brief outline of how the zero curve is followed is given here. Complete details of the algorithm and convergence proofs are in [13]. The basic idea, which has been used by several authors [7, 9, 10], is to parameterize the zero curve of $\rho_a(\lambda, x)$ by arc length and then solve an initial value problem. The zero curve $(\lambda(s), x(s))$ emanating from $(0, a)$ satisfies

$$\rho_a(\lambda(s), x(s)) = 0, \qquad \lambda(0) = 0, \qquad x(0) = a \tag{1}$$

and thus is given by the solution of the initial value problem

$$\begin{aligned} &\frac{d}{ds}\rho_a(\lambda(s), x(s)) = 0, \\ &\left\|\left(\frac{d\lambda}{ds}, \frac{dx}{ds}\right)\right\| = 1, \qquad \lambda(0) = 0, \qquad x(0) = a. \end{aligned} \tag{2}$$

Computing $(d\lambda/ds, dx/ds)$ reduces to finding the kernel of the $n \times (n + 1)$ matrix $[a - f(x), I - \lambda Df(x)]$ which always has full rank [3, 13]. ($Df(x)$ is the Jacobian matrix of $f(x)$.) Therefore there are no "singular points" along the curve.

Of course the zero curve of $\rho_a(\lambda, x)$ could be followed by a general curve tracking program like that of Kubicek [10], but since the ultimate objective is a fixed point and not the zero curve of ρ_a, special tactics are called for. There are several other reasons for not using the program in [10]. That program produces a specified number of data points on the curve, and it would be difficult to make it stop exactly at $\lambda(s) = 1$. Furthermore, the numerical linear algebra and ordinary differential equation (ODE) techniques in that program are primitive compared with current technology (as, for example, [2] and [12]).

Computing Zeros. This algorithm for computing fixed points can be used to find zeros also, but then global convergence is not guaranteed. The homotopy map for zeros is

$$\Phi_a(\lambda, x) = \lambda f(x) + (1 - \lambda)(x - a),$$

and the rest of the details are similar to the fixed point case. Observe that in the fixed point case, the zero curve of ρ_a has finite length (in fact, fairly short for practical problems), and thus stiffness of the ODE is never a problem. However, the zero curve of ϕ_a can (and in practice frequently does) wander off to infinity, and the ODE can be stiff.

There are numerous conditions on f guaranteeing that the zero curve of Φ_a reaches a zero [3, 13], but these are frequently not satisfied for practical problems. One such sufficient condition is that for some $r > 0$,

$$x^t f(x) \ge 0 \qquad \text{on} \quad \|x\| = r.$$

Then $f(x)$ has a zero in the ball $\|x\| \le r$, and for almost all a in the interior of this ball, there is a zero curve of Φ_a connecting $(0, a)$ to $(1, \bar{x})$, and the Jacobian matrix $D\Phi_a$ has full rank along this zero curve [13].

In view of the preceding comments, the computer code should be used with caution on zero finding problems.

Organizational Details

There are seven subroutines: FIXPT, FODE, DCPOSE, INNER, STEP, INTRP, and ROOT. The user need only call FIXPT; all the others are directly or indirectly used by FIXPT.

The subroutines STEP, INTRP, and ROOT are from [12]. STEP is used to solve the initial value problem, and INTRP and ROOT are used to calculate the vector $x(s)$ corresponding to $\lambda(s) = 1$. FODE specifies the ordinary differential equation for STEP. FODE must determine the kernel of a matrix, and it uses DCPOSE and INNER for this purpose. DCPOSE was taken from [2] with minor modifications.

FIXPT and FODE contain dimension statements which limit N, and also use labeled common. These restrictions could have been avoided by considerably lengthening the call lists of STEP, FODE, and FIXPT. Instead we chose to use STEP verbatim, so that the user needs only one version of STEP, and future improvements in STEP can be easily incorporated into this fixed point package. The difficulty arises because FIXPT has parameters which must be passed to FODE via STEP, and DCPOSE must return information back up through FODE and STEP to FIXPT.

FIXPT limits the number of steps to 1000, but this can be changed in the DATA statement that sets the value of LIMITD.

FIXPT and FODE contain dimension statements which limit N to 100.

STEP and ROOT use machine dependent constants, for which appropriate DATA statements must be chosen, as explained in the listing. No other modifications are required by the user.

The user must supply two subroutines, F(X, V) and FJAC (X, V, K). Subroutine F evaluates f at X and returns the result in (the vector) V. Subroutine FJAC evaluates the Kth column of the Jacobian matrix of f at X and returns the result in V. FJAC may, of course, produce finite difference approximations to the Jacobian matrix. The effect of finite difference approximations on the overall efficiency and accuracy of the algorithm has not been explored so far.

Modifications for Large Sparse Problems. FIXPT was designed for low-dimensional ($n \leq 100$) problems where the Jacobian matrix of f is dense. It is not difficult to modify FIXPT for a problem where n is very large but the Jacobian matrix of f is sparse. Write the Jacobian matrix of the homotopy map as

$$D\rho_a(x, \lambda) = [I - \lambda Df(x), a - f(x)].$$

Note that the order of the variables has been switched. The subroutine DCPOSE essentially reduces $D\rho_a(x, \lambda)$ to upper triangular form. If $Df(x)$ is sparse, then $D\rho_a(x, \lambda)$ can be reduced to upper triangular form efficiently by, e.g., plane rotations. Probably DCPOSE should be tailored to the particular structure of $Df(x)$. FODE computes the kernel of the upper triangular matrix produced by DCPOSE. This calculation in FODE would have to be changed and again should probably be tailored to the particular form of $Df(x)$. Thus FODE and DCPOSE would require major changes, but all the other subroutines would remain the same (except, of course, for the DIMENSION statements in FIXPT). It might also be desirable to reduce the storage requirements of STEP as explained in [12].

Testing. FIXPT has been tested on several hundred problems of mixed polynomial, exponential, and trigonometric types, with n ranging from 2 to 100. Some problems, for example, an exponential with a rapidly oscillating trigonometric exponent, give the method a great deal of difficulty. On the other hand, high-degree ($\geq$10) polynomials in 100 dimensions are handled easily. When used properly, the performance of FIXPT has been entirely satisfactory. The computed

Table I

n	ND	Arc length	FIXPT time	ZSYSTM time
5	52	2.71	0.5	0.5
10	74	3.73	2.3	17.3*
15	97	4.49	6.9	†
20	73	5.16	10.3	†
25	81	5.70	19.6	†
30	108	6.19	40.3	†
35	121	6.65	69.1	†
40	121	7.08	98.0	†
45	123	7.48	134.7	†
50	129	7.85	187.8	†

* Did not converge.

† No convergence, reported "singular Jacobian."

fixed points are usually accurate to within the tolerance EPS and have never been in error by more than 10 EPS. Some typical computational results are given in [13]. Shown in Table I are the results for Brown's function

$$f_1(x) = x_1 - \left(\prod_{i=1}^{n} x_i - 1\right),$$

$$f_j(x) = x_j - \left(\sum_{i=1}^{n} x_i + x_j - (n+1)\right), \qquad j = 2, \ldots, n,$$

which was suggested by R. Saigal as a particularly difficult problem (because the Jacobian matrix is ill conditioned). ZSYSTM is from the IMSL library and uses a quasi-Newton method. ND is the number of Jacobian evaluations, and the CPU time is execution time (in seconds) on a CDC 6500. EPS = 1.E−8, ARCTOL = 1.0E−3 for N = 5, 10, 20, 25, and ARCTOL = 1.0E−5 for N = 15, 30, 35, 40, 45, 50. For this problem the zero curve does not turn back, although in the next example from [13] the zero curve turns back many times. Therefore, the use of the parameter λ as a *dependent* variable is crucial.

$$f_k(x) = \exp\left(\cos\left(k \sum_{i=1}^{5} x_i\right)\right), \qquad k = 1, \ldots, 5.$$

Starting from zero, both Steffensen's method and ZSYSTM failed to find a fixed point of $f(x)$. FIXPT required 6.39 seconds of CPU time, 513 Jacobian evaluations, with an arc length of 14.83, to compute a fixed point accurate to eight places.

Referee's Note. The single precision program given in the listing was run successfully in double precision on an IBM 3033, using the AUTODBL feature of the Fortran Extended H Compiler. However, the author reports that this worked on another system only after the final argument (0.0) in the two references to INNER in DCPOSE was changed to double precision.

REFERENCES

1. Boggs, P. The solution of nonlinear systems of equations by A-stable integration techniques. *SIAM J. Numer. Anal. 8* (1971), 767–785.
2. Businger, P., and Golub, G.H. Linear least squares solutions by Householder transformations. *Numer. Math. 7* (1965), 269–276.
3. Chow, S.N., Mallet-Paret, J., and Yorke, J.A. Finding zeros of maps: Homotopy methods that are constructive with probability one. *Math. Comput. 32* (1978), 887–899.
4. Dahlquist, G., Bjorck, A., and Anderson, N. *Numerical Methods.* Prentice-Hall, Englewood Cliffs, N.J., 1974.
5. Eaves, B.C. Homotopies for the computation of fixed points. *Math. Program. 3* (1972), 1–22.
6. Eaves, B.C., and Saigal, R. Homotopies for computation of fixed points on unbounded regions. *Math. Program. 3* (1972), 225–237.
7. Keller, H.B. Numerical solution of bifurcation and nonlinear eigenvalue problems. In *Applications of Bifurcation Theory.* Academic Press, New York, 1977.
8. Kellogg, R.B., Li, T.Y., and Yorke, J. A constructive proof of the Brouwer fixed-point theorem and computational results. *SIAM J. Numer. Anal. 13* (1976), 473–483.

9. KLOPFENSTEIN, R.W. Zeros of nonlinear functions. *J. ACM 8* (1961), 336–373.
10. KUBICEK, M. Dependence of solutions of nonlinear systems on a parameter. *ACM Trans. Math. Softw. 2* (1976), 98–107.
11. ORTEGA, J.M., AND RHEINBOLDT, W.C. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York, 1970.
12. SHAMPINE, L.F., AND GORDON, M.K. *Computer Solution of Ordinary Differential Equations: The Initial Value Problem.* Freeman, San Francisco, 1975.
13. WATSON, L.T. A globally convergent algorithm for computing fixed points of C^2 maps. *Appl. Math. Comput. 5* (1979), 297–311.

ALGORITHM

```
      DIMENSION Y(1Ø1)                                                  3Ø
C                                                                       4Ø
C                                                                       5Ø
C     MAIN PROGRAM AND USER WRITTEN SUBROUTINES FOR THE FUNCTION        6Ø
C     F WHOSE KTH COMPONENT IS                                          7Ø
C        EXP(COS(K*(X(1)+X(2)+...+X(N))))                               8Ø
C                                                                       9Ø
C                                                                       1ØØ
      N=3                                                               11Ø
      ARCTOL=1.ØE-4                                                     12Ø
      EPS=1.ØE-8                                                        13Ø
      CALL SECOND(TIME1)                                                14Ø
      IFLAG=Ø                                                           15Ø
      CALL FIXPT(N,Y,ARCTOL,EPS,ARCLEN,NFE,IFLAG)                       16Ø
      NP1=N+1                                                           17Ø
      CALL SECOND(TIME2)                                                18Ø
      WRITE (6,3Ø) ARCLEN,IFLAG,(Y(L),L=1,NP1)                          19Ø
3Ø    FORMAT(/11H ARC LENGTH,1PE18.9,4X,4HCODE,I3/(1X,7E17.9))
      WRITE (6,4Ø) NFE                                                  21Ø
4Ø    FORMAT(/I8,21H JACOBIAN EVALUATIONS/)
      EXTIME=TIME2-TIME1                                                23Ø
      WRITE (6,5Ø) EXTIME                                               24Ø
5Ø    FORMAT(22H EXECUTION TIME(SEC) =,F8.2/1H1)
6Ø    CONTINUE                                                          26Ø
      STOP                                                              27Ø
      END                                                               28Ø


      SUBROUTINE SECOND(T)
C
C  **** THIS IS A DUMMY ROUTINE ****
C
C  T SHOULD BE SET EQUAL TO A COUNTER COUNTING SECONDS
C
      T = Ø.Ø
      RETURN
      END


      SUBROUTINE F(X,V)                                                 29Ø
C                                                                       
C                                                                       
C     SUBROUTINE TO EVALUATE THE FUNCTION WHOSE JTH COMPONENT IS
C         EXP(COS(J*(X(1)+...+X(N))))
C
C     ON RETURN  V  CONTAINS  F(X)
C
C
      REAL X(1ØØ),V(1ØØ)                                                3ØØ
      COMMON /FIXEDP/ YPOLD(1Ø1),A(1ØØ),NFE,N,NP1,IFLAG                 31Ø
      SUMX=Ø.Ø                                                          32Ø
      DO 2Ø J=1,N                                                       33Ø
2Ø    SUMX=SUMX+X(J)                                                    34Ø
      DO 3Ø K=1,N                                                       35Ø
3Ø    V(K)=EXP(COS(FLOAT(K)*SUMX))
      RETURN                                                            37Ø
      END                                                               38Ø


      SUBROUTINE FJAC(X,V,K)                                            39Ø
C
C
```

```
C     EVALUATES THE JACOBIAN MATRIX OF THE FUNCTION WHOSE JTH
C     COMPONENT IS
C         EXP(COS(J*(X(1)+...+X(N))))
C
C     ON RETURN  V  CONTAINS THE KTH COLUMN OF THE JACOBIAN MATRIX
C     DF(X) .
C
C
C
      REAL X(1ØØ),V(1ØØ)                                              4ØØ
      COMMON /FIXEDP/ YPOLD(1Ø1),A(1ØØ),NFE,N,NP1,IFLAG               41Ø
      IF (K .GT. 1) RETURN                                            42Ø
      SUMX=Ø.Ø                                                        43Ø
      DO 3Ø J=1,N                                                     44Ø
3Ø    SUMX=SUMX+X(J)                                                  45Ø
4Ø    DO 6Ø J=1,N                                                     46Ø
      FJ=FLOAT(J)
      ECKX=EXP(COS(FJ*SUMX))
6Ø    V(J)=-FJ*SIN(FJ*SUMX)*ECKX
      RETURN                                                          49Ø
C                                                                     5ØØ
C                                                                     51Ø
C     END OF TEST PROGRAMS.                                           52Ø
C                                                                     53Ø
C  ***************************************************************  54Ø
C                                                                     55Ø
      END                                                             56Ø


      SUBROUTINE FIXPT(N,Y,ARCTOL,EPS,ARCLEN,NFE,IFLAG)               1Ø
C                                                                     2Ø
C SUBROUTINE  FIXPT  FINDS A FIXED POINT OR ZERO OF THE               3Ø
C N-DIMENSIONAL VECTOR FUNCTION F(X).  FOR THE FIXED POINT            4Ø
C PROBLEM F(X) IS ASSUMED TO BE A C2 MAP OF SOME BALL INTO            5Ø
C ITSELF.  THE EQUATION  X = F(X)  IS SOLVED BY                       6Ø
C FOLLOWING THE ZERO CURVE OF THE HOMOTOPY MAP                        7Ø
C                                                                     8Ø
C  LAMBDA*(X - F(X)) + (1 - LAMBDA)*(X - A)  ,                        9Ø
C                                                                     1ØØ
C STARTING FROM LAMBDA = Ø, X = A.  THE CURVE IS PARAMETERIZED        11Ø
C BY ARC LENGTH S, AND IS FOLLOWED BY SOLVING THE ORDINARY            12Ø
C DIFFERENTIAL EQUATION  D(HOMOTOPY MAP)/DS = Ø  FOR                  13Ø
C Y(S) = (LAMBDA(S), X(S)).                                           14Ø
C                                                                     15Ø
C FOR THE ZERO FINDING PROBLEM F(X) IS ASSUMED TO BE A C2 MAP         16Ø
C SUCH THAT FOR SOME R .GT. Ø,  X*F(X) .GE. Ø  WHENEVER NORM(X) = R.  17Ø
C THE EQUATION  F(X) = Ø  IS SOLVED BY FOLLOWING THE ZERO CURVE       18Ø
C OF THE HOMOTOPY MAP                                                 19Ø
C                                                                     2ØØ
C   LAMBDA*F(X) + (1 - LAMBDA)*(X - A)                                21Ø
C                                                                     22Ø
C EMANATING FROM LAMBDA = Ø, X = A.                                   23Ø
C                                                                     24Ø
C  A  MUST BE AN INTERIOR POINT OF THE ABOVE MENTIONED BALLS.         25Ø
C                                                                     26Ø
C THIS CODE IS BASED ON THE ALGORITHM IN L. T. WATSON, A              27Ø
C GLOBALLY CONVERGENT ALGORITHM FOR COMPUTING FIXED POINTS OF         28Ø
C C2 MAPS, APPL. MATH. COMPUT., 1978.                                 29Ø
C                                                                     3ØØ
C                                                                     31Ø
C THE USER MUST SUPPLY A SUBROUTINE  F(X,V)  WHICH EVALUATES          32Ø
C F(X) AT X AND RETURNS THE VECTOR F(X) IN V, AND A SUBROUTINE        33Ø
C FJAC(X,V,K)  WHICH RETURNS IN V THE KTH COLUMN OF THE               34Ø
C JACOBIAN MATRIX OF F(X) EVALUATED AT X.  FIXPT  DIRECTLY OR         35Ø
C INDIRECTLY USES THE SUBROUTINES  STEP , INTRP , ROOT , FODE ,       36Ø
C F , FJAC , DCPOSE , AND THE FUNCTION  INNER .  FIXPT  AND           37Ø
C FODE  CONTAIN DIMENSION STATEMENTS WHICH LIMIT N TO 1ØØ.            38Ø
C STEP  AND  ROOT  CONTAIN MACHINE DEPENDENT CONSTANTS.  NO           39Ø
C OTHER MODIFICATIONS BY THE USER ARE REQUIRED.                       4ØØ
C                                                                     41Ø
C                                                                     42Ø
C ON INPUT:                                                           43Ø
C                                                                     44Ø
C N  IS THE DIMENSION OF X AND F(X).                                  45Ø
```

```
C                                                                      46Ø
C Y  IS AN ARRRAY OF LENGTH  N + 1.  (Y(2),...,Y(N+1)) = A  IS THE     47Ø
C    STARTING POINT FOR THE ZERO CURVE.                                48Ø
C                                                                      49Ø
C ARCTOL  IS THE LOCAL ERROR ALLOWED THE ODE SOLVER WHEN               5ØØ
C    FOLLOWING THE ZERO CURVE.  IF  ARCTOL .LE. Ø.Ø  ON INPUT          51Ø
C    IT IS RESET TO  .5*SQRT(EPS).  NORMALLY  ARCTOL  SHOULD           52Ø
C    BE CONSIDERABLY LARGER THAN EPS.                                  53Ø
C                                                                      54Ø
C EPS  IS THE LOCAL ERROR ALLOWED THE ODE SOLVER WHEN VERY             55Ø
C    NEAR THE FIXED POINT(ZERO).  EPS  IS APPROXIMATELY THE            56Ø
C    ABSOLUTE ERROR IN THE COMPUTED FIXED POINT(ZERO).                 57Ø
C                                                                      58Ø
C IFLAG  CAN BE -1, Ø, 2, OR 3.  IFLAG  SHOULD BE Ø ON THE FIRST       59Ø
C    CALL TO  FIXPT  FOR THE PROBLEM  X=F(X), AND -1 FOR THE           6ØØ
C    PROBLEM  F(X)=Ø.  IN CERTAIN SITUATIONS  IFLAG  IS SET TO         61Ø
C    2 OR 3 BY  FIXPT, AND  FIXPT  CAN BE CALLED AGAIN WITHOUT         62Ø
C    CHANGING  IFLAG.                                                  63Ø
C                                                                      64Ø
C Y, ARCTOL, EPS, ARCLEN, NFE, AND IFLAG SHOULD ALL BE                 65Ø
C VARIABLES IN THE CALLING PROGRAM.                                    66Ø
C                                                                      67Ø
C                                                                      68Ø
C ON OUTPUT:                                                           69Ø
C                                                                      7ØØ
C N  IS UNCHANGED.                                                     71Ø
C                                                                      72Ø
C Y(1) = LAMBDA, (Y(2),...,Y(N+1)) = X, AND Y IS AN APPROXIMATE        73Ø
C    ZERO OF THE HOMOTOPY MAP.  NORMALLY LAMBDA = 1 AND X IS A         74Ø
C    FIXED POINT(ZERO) OF F(X).  IN ABNORMAL SITUATIONS LAMBDA         75Ø
C    MAY ONLY BE NEAR 1 AND X IS NEAR A FIXED POINT(ZERO).             76Ø
C                                                                      77Ø
C ARCTOL = EPS AFTER A NORMAL RETURN (IFLAG = 1).                      78Ø
C                                                                      79Ø
C EPS  IS UNCHANGED AFTER A NORMAL RETURN (IFLAG = 1).  IT IS          8ØØ
C    INCREASED TO AN APPROPRIATE VALUE ON THE RETURN IFLAG = 2.        81Ø
C                                                                      82Ø
C ARCLEN  IS THE LENGTH OF THE PATH FOLLOWED.                          83Ø
C                                                                      84Ø
C NFE  IS THE NUMBER OF FUNCTION EVALUATIONS (= NUMBER OF              85Ø
C    JACOBIAN EVALUATIONS).                                            86Ø
C                                                                      87Ø
C IFLAG =                                                              88Ø
C  -1   CAUSES  FIXPT  TO INITIALIZE EVERYTHING FOR THE PROBLEM        89Ø
C       F(X) = Ø (USE ON FIRST CALL).                                  9ØØ
C                                                                      91Ø
C   Ø   CAUSES  FIXPT  TO INITIALIZE EVERYTHING FOR THE PROBLEM        92Ø
C       X = F(X) (USE ON FIRST CALL).                                  93Ø
C                                                                      94Ø
C   1   NORMAL RETURN.                                                 95Ø
C                                                                      96Ø
C   2   SPECIFIED ERROR TOLERANCE CANNOT BE MET.  EPS HAS BEEN         97Ø
C       INCREASED TO A SUITABLE VALUE.  TO CONTINUE, JUST CALL         98Ø
C       FIXPT  AGAIN WITHOUT CHANGING ANY PARAMETERS.                  99Ø
C                                                                      1ØØØ
C   3   STEP  HAS BEEN CALLED 1ØØØ TIMES.  TO CONTINUE, CALL          1Ø1Ø
C       FIXPT  AGAIN WITHOUT CHANGING ANY PARAMETERS.                  1Ø2Ø
C                                                                      1Ø3Ø
C   4   JACOBIAN MATRIX DOES NOT HAVE FULL RANK.  THE ALGORITHM        1Ø4Ø
C       HAS FAILED (THE ZERO CURVE OF THE HOMOTOPY MAP CANNOT BE       1Ø5Ø
C       FOLLOWED ANY FURTHER).                                         1Ø6Ø
C                                                                      1Ø7Ø
C   5   EPS  (OR  ARCTOL ) IS TOO LARGE.  THE PROBLEM SHOULD BE       1Ø8Ø
C       RESTARTED BY CALLING  FIXPT  WITH A SMALLER  EPS  (OR          1Ø9Ø
C       ARCTOL ) AND  IFLAG = Ø (-1).                                  11ØØ
C                                                                      111Ø
C   6   I - DF(X)  IS NEARLY SINGULAR AT THE FIXED POINT(DF(X) IS      112Ø
C       NEARLY SINGULAR AT THE ZERO).  ANSWER MAY NOT BE               113Ø
C       ACCURATE.                                                      114Ø
C                                                                      115Ø
C   7   ILLEGAL INPUT PARAMETERS, A FATAL ERROR.                       116Ø
```

```
C                                                                       117Ø
C *****  ARRAY DECLARATIONS.  *****                                     118Ø
C                                                                       119Ø
C THE FOLLOWING ARRAYS ARE FOR A MAXIMUM OF  N = 1ØØ VARIABLES.         12ØØ
C TO HANDLE UP TO  N  VARIABLES, CHANGE EACH 1ØØ AND 1Ø1 TO  N          121Ø
C AND  N + 1  RESPECTIVELY.                                             122Ø
C                                                                       123Ø
C ARRAYS NEEDED BY THE ODE SUBROUTINE  STEP .                           124Ø
      REAL Y(1Ø1),WT(1Ø1),PHI(1Ø1,16),P(1Ø1),YP(1Ø1),PSI(12)            125Ø
C                                                                       126Ø
      COMMON /FIXEDP/ YPOLD(1Ø1),A(1ØØ),NFEC,NC,NP1,IFLAGC             127Ø
C                                                                       128Ø
C *****  END OF DIMENSIONAL INFORMATION.  *****                         129Ø
C                                                                       13ØØ
      EXTERNAL FODE                                                     131Ø
      LOGICAL START,CRASH,ST99                                          132Ø
C                                                                       133Ø
C LIMITD  IS AN UPPER BOUND ON THE NUMBER OF STEPS.  IT MAY BE          134Ø
C CHANGED BY CHANGING THE FOLLOWING DATA STATEMENT:                     135Ø
      DATA LIMITD/1ØØØ/                                                 136Ø
C                                                                       137Ø
C                                                                       138Ø
C :  :  :  :  :  :  :  :  :  :  :  :  :  :  :  :  :  :  :  :  :  :     139Ø
      IF (N .LE. Ø  .OR.  EPS .LE. Ø.Ø) IFLAG=7                         14ØØ
      IF (IFLAG .LE. Ø) GO TO 1Ø                                        141Ø
      IF (IFLAG .EQ. 2) GO TO 35                                        142Ø
      IF (IFLAG .EQ. 3) GO TO 3Ø                                        143Ø
C ONLY VALID INPUT FOR  IFLAG  IS -1, Ø, 2, 3.                          144Ø
      IFLAG=7                                                           145Ø
      RETURN                                                            146Ø
C                                                                       147Ø
C *****  INITIALIZATION BLOCK.  *****                                   148Ø
C                                                                       149Ø
1Ø    ARCLEN=Ø.Ø                                                        15ØØ
      S=Ø.Ø                                                             151Ø
      IF (ARCTOL .LE. Ø.Ø) ARCTOL=.5*SQRT(EPS)                          152Ø
      NFEC=Ø                                                            153Ø
      NC=N                                                              154Ø
      IFLAGC=IFLAG                                                      155Ø
      NP1=N+1                                                           156Ø
      SQNP1=SQRT(FLOAT(NP1))                                            157Ø
      CURTOL=.Ø1/SQNP1                                                  158Ø
      ST99=.FALSE.                                                      159Ø
      START=.TRUE.                                                      16ØØ
      CRASH=.FALSE.                                                     161Ø
      H=.1                                                              162Ø
      EPSSTP=ARCTOL                                                     163Ø
C SET INITIAL CONDITIONS FOR ORDINARY DIFFERENTIAL EQUATION.            164Ø
      YPOLD(1)=1.Ø                                                      165Ø
      Y(1)=Ø.Ø                                                          166Ø
      WT(1)=1.Ø                                                         167Ø
      DO 2Ø J=2,NP1                                                     168Ø
        YPOLD(J)=Ø.Ø                                                    169Ø
        A(J-1)=Y(J)                                                     17ØØ
        WT(J)=1.Ø                                                       171Ø
2Ø    CONTINUE                                                          172Ø
3Ø    LIMIT=LIMITD                                                      173Ø
C                                                                       174Ø
C *****  END OF INITIALIZATION BLOCK.  *****                            175Ø
C                                                                       176Ø
C                                                                       177Ø
C *****  MAIN LOOP.  *****                                              178Ø
C                                                                       179Ø
35    DO 15Ø ITER=1,LIMIT                                               18ØØ
      IF (Y(1) .GE. Ø.Ø) GO TO 5Ø                                       181Ø
4Ø    IFLAG=5                                                           182Ø
      RETURN                                                            183Ø
5Ø    IF (S .LE. 7.Ø*SQNP1) GO TO 8Ø                                    184Ø
C ARC LENGTH IS GETTING TOO LONG, THE PROBLEM WILL BE                   185Ø
C RESTARTED WITH A DIFFERENT  A  VECTOR.                                186Ø
      ARCLEN=ARCLEN+S                                                   187Ø
      S=Ø.Ø                                                             188Ø
6Ø    START=.TRUE.                                                      189Ø
      CRASH=.FALSE.                                                     19ØØ
C COMPUTE A NEW  A  VECTOR.                                             191Ø
      CALL F(Y(2),YP)                                                   192Ø
```

```
      DO 7Ø JW=1,N                                                      193Ø
        AOLD=A(JW)                                                      194Ø
        IF (IFLAGC .LT. Ø)                                              195Ø
     +  A(JW)=Y(1)*YP(JW)/(1.Ø - Y(1)) + Y(JW+1)                        196Ø
        IF (IFLAGC .EQ. Ø)                                              197Ø
     +  A(JW)=(Y(JW+1) - Y(1)*YP(JW))/(1.Ø - Y(1))                      198Ø
        IF (ABS(A(JW)-AOLD) .GT. .95) GO TO 4Ø                          199Ø
7Ø    CONTINUE                                                          2ØØØ
      GO TO 1ØØ                                                         2Ø1Ø
8Ø    IF (Y(1) .LE. .99  .OR. ST99) GO TO 1ØØ                           2Ø2Ø
C WHEN LAMBDA REACHES .99, THE PROBLEM WILL BE RESTARTED WITH           2Ø3Ø
C A NEW  A  VECTOR.                                                     2Ø4Ø
9Ø    ST99=.TRUE.                                                       2Ø5Ø
      EPSSTP=EPS                                                        2Ø6Ø
      ARCTOL=EPS                                                        2Ø7Ø
      GO TO 6Ø                                                          2Ø8Ø
C                                                                       2Ø9Ø
C TAKE A STEP ALONG THE CURVE.                                          21ØØ
1ØØ   CALL STEP(S,Y,FODE,NP1,H,EPSSTP,WT,START,HOLD,K,KOLD,CRASH,       211Ø
     +          PHI,P,YP,PSI)                                           212Ø
      NFE=NFEC                                                          213Ø
C CHECK IF THE STEP WAS SUCCESSFUL.                                     214Ø
      IF (IFLAGC .NE. 4) GO TO 12Ø                                      215Ø
      IFLAG=4                                                           216Ø
      RETURN                                                            217Ø
12Ø   IF (.NOT. CRASH) GO TO 13Ø                                        218Ø
C RETURN CODE FOR ERROR TOLERANCE TOO SMALL.                            219Ø
      IFLAG=2                                                           22ØØ
C CHANGE ERROR TOLERANCES.                                              221Ø
      EPS=EPSSTP                                                        222Ø
      IF (ARCTOL .LT. EPS) ARCTOL=EPS                                   223Ø
C CHANGE LIMIT ON NUMBER OF ITERATIONS.                                 224Ø
      LIMS=LIMIT+ITER                                                   225Ø
      GO TO 22Ø                                                         226Ø
C                                                                       227Ø
13Ø   EPSSTP=ARCTOL                                                     228Ø
      IF (ABS(YP(1)) .LE. CURTOL) EPSSTP=EPS                            229Ø
      IF (Y(1) .LT. 1.Ø) GO TO 15Ø                                      23ØØ
      IF (ST99) GO TO 16Ø                                               231Ø
C                                                                       232Ø
C IF LAMBDA .GE. 1.Ø BUT THE PROBLEM HAS NOT BEEN RESTARTED             233Ø
C WITH A NEW  A  VECTOR, BACK UP AND RESTART.                           234Ø
C                                                                       235Ø
      S99=S-.5*HOLD                                                     236Ø
C GET AN APPROXIMATE ZERO Y(S) WITH  Y(1)=LAMBDA .LT. 1.Ø               237Ø
135   CALL INTRP(S,Y,S99,WT,P,NP1,KOLD,PHI,PSI)                         238Ø
      IF (WT(1) .LT. 1.Ø) GO TO 14Ø                                     239Ø
      S99=.5*(S-HOLD+S99)                                               24ØØ
      GO TO 135                                                         241Ø
C                                                                       242Ø
14Ø   DO 144 JUDY=1,NP1                                                 243Ø
        Y(JUDY)=WT(JUDY)                                                244Ø
        YPOLD(JUDY)=P(JUDY)                                             245Ø
144   WT(JUDY)=1.Ø                                                      246Ø
      S=S99                                                             247Ø
      GO TO 9Ø                                                          248Ø
C                                                                       249Ø
15Ø   CONTINUE                                                          25ØØ
C                                                                       251Ø
C *****  END OF MAIN LOOP.  *****                                       252Ø
C                                                                       253Ø
C LAMBDA HAS NOT REACHED 1 IN 1ØØØ STEPS.                               254Ø
      IFLAG=3                                                           255Ø
      RETURN                                                            256Ø
C                                                                       257Ø
C                                                                       258Ø
C USE INVERSE INTERPOLATION TO GET THE ANSWER AT LAMBDA = 1.Ø .         259Ø
C                                                                       26ØØ
16Ø   SA=S-HOLD                                                         261Ø
      SB=S                                                              262Ø
      LCODE=1                                                           263Ø
17Ø   CALL ROOT(SOUT,Y1SOUT,SA,SB,EPS,EPS,LCODE)                        264Ø
C ROOT  FINDS S SUCH THAT Y(1)(S) = LAMBDA = 1 .                        265Ø
      IF (LCODE .GT. Ø) GO TO 19Ø                                       266Ø
      CALL INTRP(S,Y,SOUT,WT,P,NP1,KOLD,PHI,PSI)                        267Ø
      Y1SOUT=WT(1)-1.Ø                                                  268Ø
```

```
      GO TO 17Ø                                                          269Ø
19Ø   IFLAG=1                                                            27ØØ
C SET IFLAG = 6 IF  ROOT  COULD NOT GET  LAMBDA = 1.Ø                    271Ø
      IF (LCODE .GT. 2) IFLAG=6                                          272Ø
      ARCLEN=ARCLEN+SA                                                   273Ø
C LAMBDA(SA) = 1.Ø .                                                     274Ø
      CALL INTRP(S,Y,SA,WT,P,NP1,KOLD,PHI,PSI)                           275Ø
C                                                                        276Ø
      DO 21Ø J=1,NP1                                                     277Ø
21Ø   Y(J)=WT(J)                                                         278Ø
      RETURN                                                             279Ø
22Ø   LIMIT=LIMS
      RETURN
      END                                                                28ØØ


      SUBROUTINE FODE(S,Y,YP)
C                                                                        282Ø
C SUBROUTINE  FODE  IS USED BY SUBROUTINE  STEP  TO SPECIFY THE          283Ø
C ORDINARY DIFFERENTIAL EQUATION  DY/DS = G(S,Y) , WHOSE SOLUTION        284Ø
C IS THE ZERO CURVE OF THE HOMOTOPY MAP.  S = ARC LENGTH,                285Ø
C YP = DY/DS, AND  Y(S) = (LAMBDA(S), X(S)) .                            286Ø
C                                                                        287Ø
C *****  ARRAY DECLARATIONS.  *****                                      288Ø
C                                                                        289Ø
C THE FOLLOWING ARRAYS ARE FOR A MAXIMUM OF N = 1ØØ VARIABLES.           29ØØ
C TO HANDLE UP TO  N  VARIABLES, CHANGE EACH 1ØØ AND 1Ø1 TO  N           291Ø
C AND  N + 1  RESPECTIVELY.                                              292Ø
C                                                                        293Ø
      REAL Y(1),YP(1)                                                    294Ø
C ARRAYS FOR COMPUTING THE JACOBIAN MATRIX AND ITS KERNAL.               295Ø
      REAL QR(1ØØ,1Ø1),ALPHA(1ØØ),TZ(1Ø1)                                296Ø
      INTEGER PIVOT(1Ø1)                                                 297Ø
C                                                                        298Ø
      REAL INNER                                                         299Ø
      COMMON /FIXEDP/ YPOLD(1Ø1),A(1ØØ),NFE,N,NP1,IFLAG                  3ØØØ
      NDIM=1ØØ                                                           3Ø1Ø
C                                                                        3Ø2Ø
C *****  END OF DIMENSIONAL INFORMATION.  *****                          3Ø3Ø
C                                                                        3Ø4Ø
      NFE=NFE+1                                                          3Ø5Ø
C NFE CONTAINS THE NUMBER OF JACOBIAN EVALUATIONS.                       3Ø6Ø
C   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *    3Ø7Ø
C                                                                        3Ø8Ø
C COMPUTE THE JACOBIAN MATRIX, STORE IT IN QR.                           3Ø9Ø
C                                                                        31ØØ
C      QR = ( A - F(X), I - LAMBDA*DF(X) )  .                            311Ø
C                                                                        312Ø
      CALL F(Y(2),TZ)                                                    313Ø
      IF (IFLAG .LT. Ø) GO TO 14Ø                                        314Ø
      DO 1ØØ J=1,N                                                       315Ø
1ØØ   QR(J,1)=A(J)-TZ(J)                                                 316Ø
      DO 12Ø K=1,N                                                       317Ø
        CALL FJAC(Y(2),TZ,K)                                             318Ø
        KP1=K+1                                                          319Ø
        DO 11Ø J=1,N                                                     32ØØ
11Ø     QR(J,KP1)=-Y(1)*TZ(J)                                            321Ø
12Ø   QR(K,KP1)=1.Ø+QR(K,KP1)                                            322Ø
      GO TO 1Ø                                                           323Ø
C                                                                        324Ø
C   QR = ( F(X) - X + A, LAMBDA*DF(X) + (1 - LAMBDA)*I ) .               325Ø
C                                                                        326Ø
14Ø   DO 15Ø J=1,N                                                       327Ø
15Ø   QR(J,1)=TZ(J)-Y(J+1)+A(J)                                          328Ø
      DO 17Ø K=1,N                                                       329Ø
        CALL FJAC(Y(2),TZ,K)                                             33ØØ
        KP1=K+1                                                          331Ø
        DO 16Ø J=1,N                                                     332Ø
16Ø     QR(J,KP1)=Y(1)*TZ(J)                                             333Ø
17Ø   QR(K,KP1)=1.Ø-Y(1)+QR(K,KP1)                                       334Ø
C                                                                        335Ø
C   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *    336Ø
C REDUCE THE JACOBIAN MATRIX TO UPPER TRIANGULAR FORM.                   337Ø
1Ø    CALL DCPOSE(NDIM,N,QR,ALPHA,PIVOT,IERR,TZ,YP)                      338Ø
      IF (IERR .EQ. Ø) GO TO 2Ø                                          339Ø
      IFLAG=4                                                            34ØØ
```

```
      RETURN                                                            341Ø
C COMPUTE KERNAL OF JACOBIAN, WHICH SPECIFIES YP=DY/DS.                 342Ø
2Ø    TZ(NP1)=1.Ø                                                       343Ø
      DO 4Ø LW=1,N                                                      344Ø
        I=NP1-LW                                                        345Ø
        IK=I+1                                                          346Ø
        SUM=Ø.Ø                                                         347Ø
        DO 3Ø J=IK,NP1                                                  348Ø
3Ø      SUM=SUM+QR(I,J)*TZ(J)                                           349Ø
4Ø    TZ(I)=-SUM/ALPHA(I)                                               35ØØ
      YPNORM=SQRT(INNER(1,NP1,TZ,TZ,Ø.Ø))                               351Ø
      DO 6Ø K=1,NP1                                                     352Ø
        KPIV=PIVOT(K)
6Ø    YP(KPIV)=TZ(K)/YPNORM                                             353Ø
      IF (INNER(1,NP1,YP,YPOLD,Ø.Ø) .GE. Ø.Ø) GO TO 8Ø                  354Ø
      DO 7Ø I=1,NP1                                                     355Ø
7Ø    YP(I)=-YP(I)                                                      356Ø
C                                                                       357Ø
C SAVE CURRENT DERIVATIVE (= TANGENT VECTOR) IN  YPOLD                  358Ø
8Ø    DO 9Ø I=1,NP1                                                     359Ø
9Ø    YPOLD(I)=YP(I)                                                    36ØØ
      RETURN                                                            361Ø
      END                                                               362Ø


      REAL FUNCTION INNER(K,M,A,B,C)                                    377Ø
      DIMENSION A(1),B(1)                                               378Ø
      SUM=C                                                             379Ø
      DO 1Ø I=K,M                                                       38ØØ
1Ø    SUM=SUM+A(I)*B(I)                                                 381Ø
      INNER=SUM                                                         382Ø
      RETURN                                                            383Ø
      END                                                               384Ø


      SUBROUTINE DCPOSE(NDIM,N,QR,ALPHA,PIVOT,IERR,Y,SUM)               385Ø
C                                                                       386Ø
C     SUBROUTINE  DCPOSE  IS A MODIFICATION OF THE ALGOL PROCEDURE      387Ø
C     DECOMPOSE  IN P. BUSINGER AND G. H. GOLUB, LINEAR LEAST           388Ø
C     SQUARES SOLUTIONS BY HOUSEHOLDER TRANSFORMATIONS,                 389Ø
C     NUMER. MATH. 7 (1965) 269-276.                                    39ØØ
C                                                                       391Ø
      INTEGER NDIM,N,PIVOT(1)                                           392Ø
      REAL QR(NDIM,1),ALPHA(N)                                          393Ø
      INTEGER IERR,I,J,JBAR,K                                           394Ø
      REAL BETA,SIGMA,ALPHAK,QRKK,Y(1),SUM(1)                           395Ø
      REAL INNER                                                        396Ø
      IERR=Ø                                                            397Ø
      NP1=N+1                                                           398Ø
      DO 2Ø J=1,NP1                                                     399Ø
        SUM(J)=INNER(1,N,QR(1,J),QR(1,J),Ø.Ø)                           4ØØØ
2Ø    PIVOT(J)=J                                                        4Ø1Ø
      DO 5ØØ K=1,N                                                      4Ø2Ø
        SIGMA=SUM(K)                                                    4Ø3Ø
        JBAR=K                                                          4Ø4Ø
        KP1=K+1                                                         4Ø5Ø
        DO 4Ø J=KP1,NP1                                                 4Ø6Ø
          IF (SIGMA .GE. SUM(J)) GO TO 4Ø                               4Ø7Ø
          SIGMA=SUM(J)                                                  4Ø8Ø
          JBAR=J                                                        4Ø9Ø
4Ø      CONTINUE                                                        41ØØ
        IF (JBAR .EQ. K) GO TO 7Ø                                       411Ø
        I=PIVOT(K)                                                      412Ø
        PIVOT(K)=PIVOT(JBAR)                                            413Ø
        PIVOT(JBAR)=I                                                   414Ø
        SUM(JBAR)=SUM(K)                                                415Ø
        SUM(K)=SIGMA                                                    416Ø
        DO 5Ø I=1,N                                                     417Ø
          SIGMA=QR(I,K)                                                 418Ø
          QR(I,K)=QR(I,JBAR)                                            419Ø
          QR(I,JBAR)=SIGMA                                              42ØØ
5Ø      CONTINUE                                                        421Ø
```

```
C       END OF COLUMN INTERCHANGE.                                      422Ø
7Ø      SIGMA=INNER(K,N,QR(1,K),QR(1,K),Ø.Ø)                            423Ø
        IF (SIGMA .NE. Ø.Ø) GO TO 6Ø                                    424Ø
        IERR=1                                                          425Ø
        RETURN                                                          426Ø
6Ø      IF (K .EQ. N) GO TO 5ØØ                                         427Ø
        QRKK=QR(K,K)                                                    428Ø
        ALPHAK=-SQRT(SIGMA)                                             429Ø
        IF (QRKK .LT. Ø.Ø) ALPHAK=-ALPHAK                               43ØØ
        ALPHA(K)=ALPHAK                                                 431Ø
        BETA=1.Ø/(SIGMA-QRKK*ALPHAK)                                    432Ø
        QR(K,K)=QRKK-ALPHAK                                             433Ø
        DO 8Ø J=KP1,NP1                                                 434Ø
8Ø      Y(J)=BETA*INNER(K,N,QR(1,K),QR(1,J),Ø.Ø)                        435Ø
        DO 1ØØ J=KP1,NP1                                                436Ø
          DO 9Ø I=K,N                                                   437Ø
            QR(I,J)=QR(I,J)-QR(I,K)*Y(J)                                438Ø
9Ø        CONTINUE                                                      439Ø
          SUM(J)=SUM(J)-QR(K,J)**2                                      44ØØ
1ØØ     CONTINUE                                                        441Ø
5ØØ   CONTINUE                                                         442Ø
      ALPHA(N)=QR(N,N)                                                  443Ø
      RETURN                                                            444Ø
      END                                                               445Ø


      SUBROUTINE STEP(X,Y,F,NEQN,H,EPS,WT,START,                        446Ø
     1  HOLD,K,KOLD,CRASH,PHI,P,YP,PSI)                                 447Ø
C                                                                       448Ø
C  SUBROUTINE  STEP  INTEGRATES A SYSTEM OF FIRST ORDER ORDINARY        449Ø
C  DIFFERENTIAL EQUATIONS ONE STEP  NORMALLY FROM X TO X+H, USING A     45ØØ
C  MODIFIED DIVIDED DIFFERENCE FORM OF THE ADAMS PECE FORMULAS.  LOCAL  451Ø
C  EXTRAPOLATION IS USED TO IMPROVE ABSOLUTE STABILITY AND ACCURACY.    452Ø
C  THE CODE ADJUSTS ITS ORDER AND STEP SIZE TO CONTROL THE LOCAL ERROR  453Ø
C  PER UNIT STEP IN A GENERALIZED SENSE.  SPECIAL DEVICES ARE INCLUDED  454Ø
C  TO CONTROL ROUNDOFF ERROR AND TO DETECT WHEN THE USER IS REQUESTING  455Ø
C  TOO MUCH ACCURACY.                                                   456Ø
C                                                                       457Ø
C  THIS CODE IS COMPLETELY EXPLAINED AND DOCUMENTED IN THE TEXT,        458Ø
C  COMPUTER SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS&  THE INITIAL   459Ø
C  VALUE PROBLEM BY L. F. SHAMPINE AND M. K. GORDON.                    46ØØ
C                                                                       461Ø
C                                                                       462Ø
C  THE PARAMETERS REPRESENT&                                            463Ø
C     X -- INDEPENDENT VARIABLE                                         464Ø
C     Y(*) -- SOLUTION VECTOR AT X                                      465Ø
C     YP(*) -- DERIVATIVE OF SOLUTION VECTOR AT  X  AFTER SUCCESSFUL    466Ø
C          STEP                                                         467Ø
C     NEQN -- NUMBER OF EQUATIONS TO BE INTEGRATED                      468Ø
C     H -- APPROPRIATE STEP SIZE FOR NEXT STEP.  NORMALLY DETERMINED BY 469Ø
C          CODE                                                         47ØØ
C     EPS -- LOCAL ERROR TOLERANCE.  MUST BE VARIABLE                   471Ø
C     WT(*) -- VECTOR OF WEIGHTS FOR ERROR CRITERION                    472Ø
C     START -- LOGICAL VARIABLE SET .TRUE. FOR FIRST STEP,  .FALSE.     473Ø
C          OTHERWISE                                                    474Ø
C     HOLD -- STEP SIZE USED FOR LAST SUCCESSFUL STEP                   475Ø
C     K -- APPROPRIATE ORDER FOR NEXT STEP (DETERMINED BY CODE)         476Ø
C      KOLD -- ORDER USED FOR LAST SUCCESSFUL STEP                      477Ø
C      CRASH -- LOGICAL VARIABLE SET .TRUE. WHEN NO STEP CAN BE TAKEN,  478Ø
C           .FALSE. OTHERWISE                                           479Ø
C   THE ARRAYS  PHI, PSI  ARE REQUIRED FOR THE INTERPOLATION SUBROUTINE 48ØØ
C   INTRP.  THE ARRAY P IS INTERNAL TO THE CODE.                        481Ø
C                                                                       482Ø
C   INPUT TO STEP                                                       483Ø
C                                                                       484Ø
C      FIRST CALL --                                                    485Ø
C                                                                       486Ø
C   THE USER MUST PROVIDE STORAGE IN HIS DRIVER PROGRAM FOR ALL ARRAYS  487Ø
C   IN THE CALL LIST, NAMELY                                            488Ø
C                                                                       489Ø
C     DIMENSION Y(NEQN),WT(NEQN),PHI(NEQN,16),P(NEQN),YP(NEQN),PSI(12)  49ØØ
C                                                                       491Ø
C   THE USER MUST ALSO DECLARE START AND CRASH LOGICAL VARIABLES        492Ø
C   AND F AN EXTERNAL SUBROUTINE, SUPPLY THE SUBROUTINE F(X,Y,YP)       493Ø
C   TO EVALUATE                                                         494Ø
C      DY(I)/DX = YP(I) = F(X,Y(1),Y(2),...,Y(NEQN))                    495Ø
```

```
C   AND INITIALIZE ONLY THE FOLLOWING PARAMETERS&                          496Ø
C      X -- INITIAL VALUE OF THE INDEPENDENT VARIABLE                     497Ø
C      Y(*) -- VECTOR OF INITIAL VALUES OF DEPENDENT VARIABLES            498Ø
C      NEQN -- NUMBER OF EQUATIONS TO BE INTEGRATED                       499Ø
C      H -- NOMINAL STEP SIZE INDICATING DIRECTION OF INTEGRATION         5ØØØ
C           AND MAXIMUM SIZE OF STEP.  MUST BE VARIABLE                   5Ø1Ø
C      EPS -- LOCAL ERROR TOLERANCE PER STEP.  MUST BE VARIABLE           5Ø2Ø
C      WT(*) -- VECTOR OF NON-ZERO WEIGHTS FOR ERROR CRITERION            5Ø3Ø
C      START -- .TRUE.                                                    5Ø4Ø
C                                                                         5Ø5Ø
C   STEP REQUIRES THE L2 NORM OF THE VECTOR WITH COMPONENTS               5Ø6Ø
C   LOCAL ERROR(L)/WT(L)  BE LESS THAN EPS FOR A SUCCESSFUL STEP.  THE    5Ø7Ø
C   ARRAY WT ALLOWS THE USER TO SPECIFY AN ERROR TEST APPROPRIATE         5Ø8Ø
C   FOR HIS PROBLEM.  FOR EXAMPLE,                                        5Ø9Ø
C      WT(L) = 1.Ø  SPECIFIES ABSOLUTE ERROR,                             51ØØ
C            =ABS(Y(L))  ERROR RELATIVE TO THE MOST RECENT VALUE OF THE   511Ø
C                 L-TH COMPONENT OF THE DERIVATIVE,                       512Ø
C            = AMAX1(WT(L),ABS(Y(L)))  ERROR RELATIVE TO THE LARGEST      513Ø
C                 MAGNITUDE OF L-TH COMPONENT OBTAINED SO FAR,            514Ø
C            = ABS(Y(L))*RELERR/EPS + ABSERR/EPS SPECIFIES A MIXED        515Ø
C                 RELATIVE-ABSOLUTE TEST WHERE RELERR IS RELATIVE         516Ø
C                 ERROR, ABSERR IS ABSOLUTE ERROR AND EPS =               517Ø
C                 AMAX1(RELERR,ABSERR)                                    518Ø
C                                                                         519Ø
C      SUBSEQUENT CALLS --                                                52ØØ
C                                                                         521Ø
C   SUBROUTINE STEP IS DESIGNED SO THAT ALL INFORMATION NEEDED TO         522Ø
C   CONTINUE THE INTEGRATION, INCLUDING THE STEP SIZE H AND THE ORDER     523Ø
C   K, IS RETURNED WITH EACH STEP.  WITH THE EXCEPTION OF THE STEP        524Ø
C   SIZE, THE ERROR TOLERANCE, AND THE WEIGHTS, NONE OF THE PARAMETERS    525Ø
C   SHOULD BE ALTERED.  THE ARRAY WT MUST BE UPDATED AFTER EACH STEP      526Ø
C   TO MAINTAIN RELATIVE ERROR TESTS LIKE THOSE ABOVE.  NORMALLY THE      527Ø
C   INTEGRATION IS CONTINUED JUST BEYOND THE DESIRED ENDPOINT AND THE     528Ø
C   SOLUTION INTERPOLATED THERE WITH SUBROUTINE INTRP.  IF IT IS          529Ø
C   IMPOSSIBLE TO INTEGRATE BEYOND THE ENDPOINT, THE STEP SIZE MAY BE     53ØØ
C   REDUCED TO HIT THE ENDPOINT SINCE THE CODE WILL NOT TAKE A STEP       531Ø
C   LARGER THAN THE H INPUT.  CHANGING THE DIRECTION OF INTEGRATION,      532Ø
C   I.E., THE SIGN OF H, REQUIRES THE USER SET START = .TRUE. BEFORE      533Ø
C   CALLING STEP AGAIN.  THIS IS THE ONLY SITUATION IN WHICH START        534Ø
C   SHOULD BE ALTERED.                                                    535Ø
C                                                                         536Ø
C   OUTPUT FROM STEP                                                      537Ø
C                                                                         538Ø
C      SUCCESSFUL STEP --                                                 539Ø
C                                                                         54ØØ
C   THE SUBROUTINE RETURNS AFTER EACH SUCCESSFUL STEP WITH START AND      541Ø
C   CRASH SET .FALSE. .  X REPRESENTS THE INDEPENDENT VARIABLE            542Ø
C   ADVANCED ONE STEP OF LENGTH HOLD FROM ITS VALUE ON INPUT AND Y        543Ø
C   THE SOLUTION VECTOR AT THE NEW VALUE OF X .  ALL OTHER PARAMETERS     544Ø
C   REPRESENT INFORMATION CORRESPONDING TO THE NEW X NEEDED TO            545Ø
C   CONTINUE THE INTEGRATION.                                             546Ø
C                                                                         547Ø
C      UNSUCCESSFUL STEP --                                               548Ø
C                                                                         549Ø
C   WHEN THE ERROR TOLERANCE IS TOO SMALL FOR THE MACHINE PRECISION,      55ØØ
C   THE SUBROUTINE RETURNS WITHOUT TAKING A STEP AND CRASH = .TRUE.       551Ø
C   AN APPROPRIATE STEP SIZE AND ERROR TOLERANCE FOR CONTINUING ARE       552Ø
C   ESTIMATED AND ALL OTHER INFORMATION IS RESTORED AS UPON INPUT         553Ø
C   BEFORE RETURNING.  TO CONTINUE WITH THE LARGER TOLERANCE, THE USER    554Ø
C   JUST CALLS THE CODE AGAIN.  A RESTART IS NEITHER REQUIRED NOR         555Ø
C   DESIRABLE.                                                            556Ø
C                                                                         557Ø
      LOGICAL START,CRASH,PHASE1,NORND                                    558Ø
      DIMENSION Y(NEQN),WT(NEQN),PHI(NEQN,16),P(NEQN),YP(NEQN),PSI(12)    559Ø
      DIMENSION ALPHA(12),BETA(12),SIG(13),W(12),V(12),G(13),             56ØØ
     1  GSTR(13),TWO(13)                                                  561Ø
      EXTERNAL F                                                          562Ø
C*********************************************************************  563Ø
C*  THE ONLY MACHINE DEPENDENT CONSTANTS ARE BASED ON THE MACHINE UNIT *  564Ø
C*  ROUNDOFF ERROR U WHICH IS THE SMALLEST POSITIVE NUMBER SUCH THAT   *  565Ø
C*  1.Ø+U .GT. 1.Ø  .  THE USER MUST CALCULATE U AND INSERT           *  566Ø
C*  TWOU=2.Ø*U AND FOURU=4.Ø*U IN THE DATA STATEMENT BEFORE CALLING    *  567Ø
C*  THE CODE.  THE ROUTINE MACHIN CALCULATES U.                          568Ø
C*  FOR THE CDC 65ØØ, 66ØØ, 76ØØ --                                   *
C*    DATA TWOU,FOURU /16424ØØØØØØØØØØØØØØØØB,16434ØØØØØØØØØØØØØØØØB/
C*  FOR THE IBM 36Ø/37Ø (SINGLE PRECISION) --                          *
```

```
C*    DATA TWOU,FOURU /Z3C2ØØØØØ,Z3C4ØØØØØ/
C*  FOR THE IBM 36Ø/37Ø (DOUBLE PRECISION) --                          *
C*    DATA TWOU,FOURU /Z342ØØØØØØØØØØØØØØ,Z344ØØØØØØØØØØØØØØ/
C*  FOR THE UNIVAC 11Ø8 AND HONEYWELL 6ØØØ --                          *
C*    DATA TWOU,FOURU /2.Ø**(-25),2.Ø**(-24)/
C*  FOR THE PDP-11 --                                                  *
C*    DATA TWOU,FOURU /2.Ø**(-22),2.Ø**(-21)/
C*********************************************************************57ØØ
      DATA TWO/2.Ø,4.Ø,8.Ø,16.Ø,32.Ø,64.Ø,128.Ø,256.Ø,512.Ø,1Ø24.Ø,      571Ø
     1  2Ø48.Ø,4Ø96.Ø,8192.Ø/                                           572Ø
      DATA GSTR/Ø.5ØØ,Ø.Ø833,Ø.Ø417,Ø.Ø264,Ø.Ø188,Ø.Ø143,Ø.Ø114,Ø.ØØ936,573Ø
     1  Ø.ØØ789,Ø.ØØ679,Ø.ØØ592,Ø.ØØ524,Ø.ØØ468/                      574Ø
      DATA G(1),G(2)/1.Ø,Ø.5/,SIG(1)/1.Ø/                              575Ø
C                                                                       576Ø
C                                                                       577Ø
C       ***     BEGIN BLOCK Ø                                           578Ø
C   CHECK IF STEP SIZE OR ERROR TOLERANCE IS TOO SMALL FOR MACHINE      579Ø
C   PRECISION.  IF FIRST STEP, INITIALIZE PHI ARRAY AND ESTIMATE A      58ØØ
C   STARTING STEP SIZE.                                                 581Ø
C                  ***                                                  582Ø
C                                                                       583Ø
C   IF STEP SIZE IS TOO SMALL, DETERMINE AN ACCEPTABLE ONE              584Ø
C                                                                       585Ø
      CRASH = .TRUE.                                                    586Ø
      IF(ABS(H) .GE. FOURU*ABS(X)) GO TO 5                              587Ø
      H = SIGN(FOURU*ABS(X),H)                                          588Ø
      RETURN                                                            589Ø
    5 P5EPS = Ø.5*EPS                                                   59ØØ
C                                                                       591Ø
C   IF ERROR TOLERANCE IS TOO SMALL, INCREASE IT TO AN ACCEPTABLE VALUE 592Ø
C                                                                       593Ø
      ROUND = Ø.Ø                                                       594Ø
      DO 1Ø L = 1,NEQN                                                  595Ø
   1Ø   ROUND = ROUND + (Y(L)/WT(L))**2                                 596Ø
      ROUND = TWOU*SQRT(ROUND)                                          597Ø
      IF(P5EPS .GE. ROUND) GO TO 15                                     598Ø
      EPS = 2.Ø*ROUND*(1.Ø + FOURU)                                     599Ø
      RETURN                                                            6ØØØ
   15 CRASH = .FALSE.                                                   6Ø1Ø
      IF(.NOT.START) GO TO 99                                           6Ø2Ø
C                                                                       6Ø3Ø
C   INITIALIZE.  COMPUTE APPROPRIATE STEP SIZE FOR FIRST STEP           6Ø4Ø
C                                                                       6Ø5Ø
      CALL F(X,Y,YP)                                                    6Ø6Ø
        SUM = Ø.Ø                                                       6Ø7Ø
        DO 2Ø L = 1,NEQN                                                6Ø8Ø
        PHI(L,1) = YP(L)                                                6Ø9Ø
        PHI(L,2) = Ø.Ø                                                  61ØØ
   2Ø   SUM  = SUM + (YP(L)/WT(L))**2                                   611Ø
      SUM = SQRT(SUM)                                                   612Ø
      ABSH = ABS(H)                                                     613Ø
      IF(EPS .LT. 16.Ø*SUM*H*H) ABSH = Ø.25*SQRT(EPS/SUM)               614Ø
      H = SIGN(AMAX1(ABSH,FOURU*ABS(X)),H)                              615Ø
      HOLD = Ø.Ø                                                        616Ø
      K = 1                                                             617Ø
      KOLD = Ø                                                          618Ø
      START = .FALSE.                                                   619Ø
      PHASE1 = .TRUE.                                                   62ØØ
      NORND = .TRUE.                                                    621Ø
      IF(P5EPS .GT. 1ØØ.Ø*ROUND) GO TO 99                               622Ø
      NORND = .FALSE.                                                   623Ø
      DO 25 L = 1,NEQN                                                  624Ø
   25   PHI(L,15) = Ø.Ø                                                625Ø
   99 IFAIL = Ø                                                         626Ø
C       ***     END BLOCK Ø     ***                                     627Ø
C                                                                       628Ø
C       ***     BEGIN BLOCK 1     ***                                   629Ø
C   COMPUTE COEFFICIENTS OF FORMULAS FOR THIS STEP.  AVOID COMPUTING    63ØØ
C   THOSE QUANTITIES NOT CHANGED WHEN STEP SIZE IS NOT CHANGED.         631Ø
C                  ***                                                  632Ø
C                                                                       633Ø
  1ØØ KP1 = K+1                                                         634Ø
      KP2 = K+2                                                         635Ø
      KM1 = K-1                                                         636Ø
      KM2 = K-2                                                         637Ø
C                                                                       638Ø
```

```
C   NS IS THE NUMBER OF STEPS TAKEN WITH SIZE H, INCLUDING THE CURRENT  639Ø
C   ONE.  WHEN K.LT.NS, NO COEFFICIENTS CHANGE                          64ØØ
      IF(H .NE. HOLD) NS = Ø                                            641Ø
        NS = MINØ(NS+1,KOLD+1)                                          642Ø
      NSP1 = NS+1                                                       643Ø
      IF (K .LT. NS) GO TO 199                                          644Ø
C                                                                       645Ø
C   COMPUTE THOSE COMPONENTS OF ALPHA(*),BETA(*),PSI(*),SIG(*) WHICH    646Ø
C   ARE CHANGED                                                         647Ø
C                                                                       648Ø
      BETA(NS) = 1.Ø                                                    649Ø
      REALNS = NS                                                       65ØØ
      ALPHA(NS) = 1.Ø/REALNS                                            651Ø
      TEMP1 = H*REALNS                                                  652Ø
      SIG(NSP1) = 1.Ø                                                   653Ø
      IF(K .LT. NSP1) GO TO 11Ø                                         654Ø
      DO 1Ø5 I = NSP1,K                                                 655Ø
        IM1 = I-1                                                       656Ø
        TEMP2 = PSI(IM1)                                                657Ø
        PSI(IM1) = TEMP1                                                658Ø
        BETA(I) = BETA(IM1)*PSI(IM1)/TEMP2                              659Ø
        TEMP1 = TEMP2 + H                                               66ØØ
        ALPHA(I) = H/TEMP1                                              661Ø
        REALI = I                                                       662Ø
 1Ø5    SIG(I+1) = REALI*ALPHA(I)*SIG(I)                                663Ø
 11Ø PSI(K) = TEMP1                                                     664Ø
C                                                                       665Ø
C   COMPUTE COEFFICIENTS G(*)                                           666Ø
C                                                                       667Ø
C   INITIALIZE V(*) AND SET W(*).  G(2) IS SET IN DATA STATEMENT        668Ø
C                                                                       669Ø
      IF(NS .GT. 1) GO TO 12Ø                                           67ØØ
      DO 115 IQ = 1,K                                                   671Ø
        TEMP3 = IQ*(IQ+1)                                               672Ø
        V(IQ) = 1.Ø/TEMP3                                               673Ø
 115    W(IQ) = V(IQ)                                                   674Ø
      GO TO 14Ø                                                         675Ø
C                                                                       676Ø
C   IF ORDER WAS RAISED, UPDATE DIAGONAL PART OF V(*)                   677Ø
C                                                                       678Ø
 12Ø IF(K .LE. KOLD) GO TO 13Ø                                          679Ø
      TEMP4 = K*KP1                                                     68ØØ
      V(K) = 1.Ø/TEMP4                                                  681Ø
      NSM2 = NS-2                                                       682Ø
      IF(NSM2 .LT. 1) GO TO 13Ø                                         683Ø
      DO 125 J = 1,NSM2                                                 684Ø
        I = K-J                                                         685Ø
 125    V(I) = V(I) - ALPHA(J+1)*V(I+1)                                 686Ø
C                                                                       687Ø
C   UPDATE V(*) AND SET W(*)                                            688Ø
C                                                                       689Ø
 13Ø LIMIT1 = KP1 - NS                                                  69ØØ
      TEMP5 = ALPHA(NS)                                                 691Ø
      DO 135 IQ = 1,LIMIT1                                              692Ø
        V(IQ) = V(IQ) - TEMP5*V(IQ+1)                                   693Ø
 135    W(IQ) = V(IQ)                                                   694Ø
      G(NSP1) = W(1)                                                    695Ø
C                                                                       696Ø
C   COMPUTE THE G(*) IN THE WORK VECTOR W(*)                            697Ø
C                                                                       698Ø
 14Ø NSP2 = NS + 2                                                      699Ø
      IF(KP1 .LT. NSP2) GO TO 199                                       7ØØØ
      DO 15Ø I = NSP2,KP1                                               7Ø1Ø
        LIMIT2 = KP2 - I                                                7Ø2Ø
        TEMP6 = ALPHA(I-1)                                              7Ø3Ø
        DO 145 IQ = 1,LIMIT2                                            7Ø4Ø
 145      W(IQ) = W(IQ) - TEMP6*W(IQ+1)                                 7Ø5Ø
 15Ø    G(I) = W(1)                                                     7Ø6Ø
 199    CONTINUE                                                        7Ø7Ø
C       ***     END BLOCK 1     ***                                     7Ø8Ø
C                                                                       7Ø9Ø
C       ***     BEGIN BLOCK 2     ***                                   71ØØ
C   PREDICT A SOLUTION P(*), EVALUATE DERIVATIVES USING PREDICTED       711Ø
C   SOLUTION, ESTIMATE LOCAL ERROR AT ORDER K AND ERRORS AT ORDERS K,   712Ø
C   K-1, K-2 AS IF CONSTANT STEP SIZE WERE USED.                        713Ø
C                 ***                                                   714Ø
```

```
C                                                                       715Ø
C   CHANGE PHI TO PHI STAR                                              716Ø
C                                                                       717Ø
      IF(K .LT. NSP1) GO TO 215                                         718Ø
      DO 21Ø I = NSP1,K                                                 719Ø
        TEMP1 = BETA(I)                                                 72ØØ
        DO 2Ø5 L = 1,NEQN                                               721Ø
  2Ø5     PHI(L,I) = TEMP1*PHI(L,I)                                     722Ø
  21Ø   CONTINUE                                                        723Ø
C                                                                       724Ø
C   PREDICT SOLUTION AND DIFFERENCES                                    725Ø
C                                                                       726Ø
  215 DO 22Ø L = 1,NEQN                                                 727Ø
        PHI(L,KP2) = PHI(L,KP1)                                         728Ø
        PHI(L,KP1) = Ø.Ø                                                729Ø
  22Ø   P(L) = Ø.Ø                                                      73ØØ
      DO 23Ø J = 1,K                                                    731Ø
        I = KP1 - J                                                     732Ø
        IP1 = I+1                                                       733Ø
      TEMP2 = G(I)                                                      734Ø
        DO 225 L = 1,NEQN                                               735Ø
          P(L) = P(L) + TEMP2*PHI(L,I)                                  736Ø
  225     PHI(L,I) = PHI(L,I) + PHI(L,IP1)                              737Ø
  23Ø   CONTINUE                                                        738Ø
      IF(NORND) GO TO 24Ø                                               739Ø
      DO 235 L = 1,NEQN                                                 74ØØ
        TAU = H*P(L) - PHI(L,15)                                        741Ø
        P(L) = Y(L) + TAU                                               742Ø
  235   PHI(L,16) = (P(L) - Y(L)) -TAU                                  743Ø
      GO TO 25Ø                                                         744Ø
  24Ø DO 245 L = 1,NEQN                                                 745Ø
  245   P(L) = Y(L) + H*P(L)                                            746Ø
  25Ø XOLD = X                                                          747Ø
      X = X + H                                                         748Ø
      ABSH = ABS(H)                                                     749Ø
      CALL F(X,P,YP)                                                    75ØØ
C                                                                       751Ø
C   ESTIMATE ERRORS AT ORDERS K,K-1,K-2                                 752Ø
      ERKM2 = Ø.Ø                                                       753Ø
      ERKM1 = Ø.Ø                                                       754Ø
      ERK = Ø.Ø                                                         755Ø
      DO 265 L = 1,NEQN                                                 756Ø
        TEMP3 = 1.Ø/WT(L)                                               757Ø
        TEMP4 = YP(L) - PHI(L,1)                                        758Ø
        IF(KM2) 265,26Ø,255                                             759Ø
  255   ERKM2 = ERKM2 + ((PHI(L,KM1)+TEMP4)*TEMP3)**2                  76ØØ
  26Ø   ERKM1 = ERKM1 + ((PHI(L,K)+TEMP4)*TEMP3)**2                     761Ø
  265   ERK = ERK + (TEMP4*TEMP3)**2                                    762Ø
      IF(KM2) 28Ø,275,27Ø                                               763Ø
  27Ø ERKM2 = ABSH*SIG(KM1)*GSTR(KM2)*SQRT(ERKM2)                       764Ø
  275 ERKM1 = ABSH*SIG(K)*GSTR(KM1)*SQRT(ERKM1)                         765Ø
28Ø   TEMP5 = ABSH*SQRT(ERK)                                            766Ø
      ERR = TEMP5*(G(K)-G(KP1))                                         767Ø
      ERK = TEMP5*SIG(KP1)*GSTR(K)                                      768Ø
      KNEW = K                                                          769Ø
C                                                                       77ØØ
C   TEST IF ORDER. SHOULD BE LOWERED                                    771Ø
C                                                                       772Ø
      IF(KM2) 299,29Ø,285                                               773Ø
  285 IF(AMAX1(ERKM1,ERKM2) .LE. ERK) KNEW = KM1                        774Ø
      GO TO 299                                                         775Ø
  29Ø IF(ERKM1 .LE. Ø.5*ERK) KNEW = KM1                                 776Ø
C                                                                       777Ø
C   TEST IF STEP SUCCESSFUL                                             778Ø
C                                                                       779Ø
  299 IF(ERR .LE. EPS) GO TO 4ØØ                                        78ØØ
C       ***     END BLOCK 2      ***                                    781Ø
C                                                                       782Ø
C       ***     BEGIN BLOCK 3     ***                                   783Ø
C   THE STEP IS UNSUCCESSFUL.  RESTORE X, PHI(*,*), PSI(*)              784Ø
C   IF THIRD CONSECUTIVE FAILURE, SET ORDER TO ONE.  IF STEP FAILS MORE 785Ø
C   THAN THREE TIMES, CONSIDER AN OPTIMAL STEP SIZE.  DOUBLE ERROR      786Ø
C   TOLERANCE AND RETURN IF ESTIMATED STEP SIZE IS TOO SMALL FOR MACHINE787Ø
C   PRECISION                                                           788Ø
C                  ***                                                  789Ø
C                                                                       79ØØ
```

```
C   RESTORE X, PHI(*,*) AND PSI(*)                                        791Ø
C                                                                         792Ø
      PHASE1 = .FALSE.                                                    793Ø
      X = XOLD                                                            794Ø
      DO 31Ø I = 1,K                                                      795Ø
        TEMP1 = 1.Ø/BETA(I)                                               796Ø
        IP1 = I+1                                                         797Ø
        DO 3Ø5 L = 1,NEQN                                                 798Ø
  3Ø5     PHI(L,I) = TEMP1*(PHI(L,I) - PHI(L,IP1))                        799Ø
  31Ø   CONTINUE                                                          8ØØØ
      IF(K .LT. 2) GO TO 32Ø                                              8Ø1Ø
      DO 315 I = 2,K                                                      8Ø2Ø
  315   PSI(I-1) = PSI(I) - H                                             8Ø3Ø
C                                                                         8Ø4Ø
C   ON THIRD FAILURE, SET ORDER TO ONE.  THEREAFTER, USE OPTIMAL STEP     8Ø5Ø
C   SIZE                                                                  8Ø6Ø
C                                                                         8Ø7Ø
  32Ø IFAIL = IFAIL + 1                                                   8Ø8Ø
      TEMP2 = Ø.5                                                         8Ø9Ø
      IF(IFAIL - 3) 335,33Ø,325                                           81ØØ
  325 IF(P5EPS .LT. Ø.25*ERK) TEMP2 = SQRT(P5EPS/ERK)                     811Ø
  33Ø KNEW = 1                                                            812Ø
  335 H = TEMP2*H                                                         813Ø
      K = KNEW                                                            814Ø
      IF(ABS(H) .GE. FOURU*ABS(X)) GO TO 34Ø                              815Ø
      CRASH = .TRUE.                                                      816Ø
      H = SIGN(FOURU*ABS(X),H)                                            817Ø
      EPS = EPS + EPS                                                     818Ø
      RETURN                                                              819Ø
  34Ø GO TO 1ØØ                                                           82ØØ
C       ***     END BLOCK 3     ***                                       821Ø
C                                                                         822Ø
C       ***     BEGIN BLOCK 4     ***                                     823Ø
C   THE STEP IS SUCCESSFUL.  CORRECT THE PREDICTED SOLUTION, EVALUATE     824Ø
C   THE DERIVATIVES USING THE CORRECTED SOLUTION AND UPDATE THE           825Ø
C   DIFFERENCES.  DETERMINE BEST ORDER AND STEP SIZE FOR NEXT STEP.       826Ø
C                   ***                                                   827Ø
  4ØØ KOLD = K                                                            828Ø
      HOLD = H                                                            829Ø
C                                                                         83ØØ
C   CORRECT AND EVALUATE                                                  831Ø
C                                                                         832Ø
      TEMP1 = H*G(KP1)                                                    833Ø
      IF(NORND) CO TO 41Ø                                                 834Ø
      DO 4Ø5 L = 1,NEQN                                                   835Ø
        RHO = TEMP1*(YP(L) - PHI(L,1)) - PHI(L,16)                        836Ø
        Y(L) = P(L) + RHO                                                 837Ø
  4Ø5   PHI(L,15) = (Y(L) - P(L)) - RHO                                   838Ø
      GO TO 42Ø                                                           839Ø
  41Ø DO 415 L = 1,NEQN                                                   84ØØ
  415   Y(L) = P(L) + TEMP1*(YP(L) - PHI(L,1))                           841Ø
  42Ø CALL F(X,Y,YP)                                                      842Ø
C                                                                         843Ø
C   UPDATE DIFFERENCES FOR NEXT STEP                                      844Ø
C                                                                         845Ø
      DO 425 L = 1,NEQN                                                   846Ø
        PHI(L,KP1) = YP(L) - PHI(L,1)                                     847Ø
  425   PHI(L,KP2) = PHI(L,KP1) - PHI(L,KP2)                             848Ø
      DO 435 I = 1,K                                                      849Ø
        DO 43Ø L = 1,NEQN                                                 85ØØ
  43Ø     PHI(L,I) = PHI(L,I) + PHI(L,KP1)                                851Ø
  435   CONTINUE                                                          852Ø
C                                                                         853Ø
C   ESTIMATE ERROR AT ORDER K+1 UNLESS&                                   854Ø
C     IN FIRST PHASE WHEN ALWAYS RAISE ORDER,                             855Ø
C     ALREADY DECIDED TO LOWER ORDER,                                     856Ø
C     STEP SIZE NOT CONSTANT SO ESTIMATE UNRELIABLE                       857Ø
C                                                                         858Ø
      ERKP1 = Ø.Ø                                                         859Ø
      IF(KNEW .EQ. KM1  .OR.  K .EQ. 12) PHASE1 = .FALSE.                 86ØØ
      IF(PHASE1) GO TO 45Ø                                                861Ø
      IF(KNEW .EQ. KM1) GO TO 455                                         862Ø
      IF(KP1 .GT. NS) GO TO 46Ø                                           863Ø
      DO 44Ø L = 1,NEQN                                                   864Ø
  44Ø   ERKP1 = ERKP1 + (PHI(L,KP2)/WT(L))**2                             865Ø
      ERKP1 = ABSH*GSTR(KP1)*SQRT(ERKP1)                                  866Ø
```

```
C                                                                       867Ø
C   USING ESTIMATED ERROR AT ORDER K+1, DETERMINE APPROPRIATE ORDER     868Ø
C   FOR NEXT STEP                                                       869Ø
C                                                                       87ØØ
      IF(K .GT. 1) GO TO 445                                            871Ø
      IF(ERKP1 .GE. Ø.5*ERK) GO TO 46Ø                                  872Ø
      GO TO 45Ø                                                         873Ø
  445 IF(ERKM1 .LE. AMIN1(ERK,ERKP1)) GO TO 455                         874Ø
      IF(ERKP1 .GE. ERK  .OR.  K .EQ. 12) GO TO 46Ø                     875Ø
C                                                                       876Ø
C   HERE ERKP1 .LT. ERK .LT. AMAX1(ERKM1,ERKM2) ELSE ORDER WOULD HAVE   877Ø
C   BEEN LOWERED IN BLOCK 2.  THUS ORDER IS TO BE RAISED                878Ø
C                                                                       879Ø
C   RAISE ORDER                                                         88ØØ
C                                                                       881Ø
  45Ø K = KP1                                                           882Ø
      ERK = ERKP1                                                       883Ø
      GO TO 46Ø                                                         884Ø
C                                                                       885Ø
C   LOWER ORDER                                                         886Ø
C                                                                       887Ø
  455 K = KM1                                                           888Ø
      ERK = ERKM1                                                       889Ø
C                                                                       89ØØ
C   WITH NEW ORDER DETERMINE APPROPRIATE STEP SIZE FOR NEXT STEP        891Ø
C                                                                       892Ø
  46Ø HNEW = H + H                                                      893Ø
      IF(PHASE1) GO TO 465                                              894Ø
      IF(P5EPS .GE. ERK*TWO(K+1)) GO TO 465                             895Ø
      HNEW = H                                                          896Ø
      IF(P5EPS .GE. ERK) GO TO 465                                      897Ø
      TEMP2 = K+1                                                       898Ø
      R = (P5EPS/ERK)**(1.Ø/TEMP2)                                      899Ø
      HNEW = ABSH*AMAX1(Ø.5,AMIN1(Ø.9,R))                               9ØØØ
      HNEW = SIGN(AMAX1(HNEW,FOURU*ABS(X)),H)                           9Ø1Ø
  465 H = HNEW                                                          9Ø2Ø
      RETURN                                                            9Ø3Ø
C       ***     END BLOCK 4     ***                                     9Ø4Ø
      END                                                               9Ø5Ø


      SUBROUTINE INTRP(X,Y,XOUT,YOUT,YPOUT,NEQN,KOLD,PHI,PSI)           9Ø6Ø
C                                                                       9Ø7Ø
C   THE METHODS IN SUBROUTINE STEP APPROXIMATE THE SOLUTION NEAR X      9Ø8Ø
C   BY A POLYNOMIAL.  SUBROUTINE INTRP APPROXIMATES THE SOLUTION AT     9Ø9Ø
C   XOUT BY EVALUATING THE POLYNOMIAL THERE.  INFORMATION DEFINING THIS 91ØØ
C   POLYNOMIAL IS PASSED FROM STEP SO INTRP CANNOT BE USED ALONE.       911Ø
C                                                                       912Ø
C   THIS CODE IS COMPLETELY EXPLAINED AND DOCUMENTED IN THE TEXT,       913Ø
C   COMPUTER SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS&  THE INITIAL  914Ø
C   VALUE PROBLEM BY L. F. SHAMPINE AND M. K. GORDON.                   915Ø
C                                                                       916Ø
C   INPUT TO INTRP --                                                   917Ø
C                                                                       918Ø
C   THE USER PROVIDES STORAGE IN THE CALLING PROGRAM FOR THE ARRAYS IN  919Ø
C   THE CALL LIST                                                       92ØØ
      DIMENSION Y(NEQN),YOUT(NEQN),YPOUT(NEQN),PHI(NEQN,16),PSI(12)     921Ø
C   AND DEFINES                                                         922Ø
C     XOUT -- POINT AT WHICH SOLUTION IS DESIRED.                       923Ø
C   THE REMAINING PARAMETERS ARE DEFINED IN STEP AND PASSED TO INTRP    924Ø
C   FROM THAT SUBROUTINE                                                925Ø
C                                                                       926Ø
C   OUTPUT FROM INTRP --                                                927Ø
C                                                                       928Ø
C     YOUT(*) -- SOLUTION AT XOUT                                       929Ø
C     YPOUT(*) -- DERIVATIVE OF SOLUTION AT XOUT                        93ØØ
C   THE REMAINING PARAMETERS ARE RETURNED UNALTERED FROM THEIR INPUT    931Ø
C   VALUES.  INTEGRATION WITH STEP MAY BE CONTINUED.                    932Ø
C                                                                       933Ø
      DIMENSION G(13),W(13),RHO(13)                                     934Ø
      DATA G(1)/1.Ø/,RHO(1)/1.Ø/                                        935Ø
C                                                                       936Ø
      HI = XOUT - X                                                     937Ø
      KI = KOLD + 1                                                     938Ø
      KIP1 = KI + 1                                                     939Ø
C                                                                       94ØØ
```

```
C   INITIALIZE W(*) FOR COMPUTING G(*)                                  941Ø
C                                                                       942Ø
      DO 5 I = 1,KI                                                     943Ø
        TEMP1 = I                                                       944Ø
    5   W(I) = 1.Ø/TEMP1                                                945Ø
      TERM = Ø.Ø                                                        946Ø
C                                                                       947Ø
C   COMPUTE G(*)                                                        948Ø
C                                                                       949Ø
      DO 15 J = 2,KI                                                    95ØØ
        JM1 = J - 1                                                     951Ø
        PSIJM1 = PSI(JM1)                                               952Ø
        GAMMA = (HI + TERM)/PSIJM1                                      953Ø
        ETA = HI/PSIJM1                                                 954Ø
        LIMIT1 = KIP1 - J                                               955Ø
        DO 1Ø I = 1,LIMIT1                                              956Ø
   1Ø     W(I) = GAMMA*W(I) - ETA*W(I+1)                                957Ø
        G(J) = W(1)                                                     958Ø
        RHO(J) = GAMMA*RHO(JM1)                                         959Ø
   15   TERM = PSIJM1                                                   96ØØ
C                                                                       961Ø
C   INTERPOLATE                                                         962Ø
C                                                                       963Ø
      DO 2Ø L = 1,NEQN                                                  964Ø
        YPOUT(L) = Ø.Ø                                                  965Ø
   2Ø   YOUT(L) = Ø.Ø                                                   966Ø
      DO 3Ø J = 1,KI                                                    967Ø
        I = KIP1 - J                                                    968Ø
        TEMP2 = G(I)                                                    969Ø
        TEMP3 = RHO(I)                                                  97ØØ
        DO 25 L = 1,NEQN                                                971Ø
          YOUT(L) = YOUT(L) + TEMP2*PHI(L,I)                            972Ø
   25     YPOUT(L) = YPOUT(L) + TEMP3*PHI(L,I)                          973Ø
   3Ø   CONTINUE                                                        974Ø
      DO 35 L = 1,NEQN                                                  975Ø
   35   YOUT(L) = Y(L) + HI*YOUT(L)                                     976Ø
      RETURN                                                            977Ø
      END                                                               978Ø


      SUBROUTINE ROOT(T,FT,B,C,RELERR,ABSERR,IFLAG)                     979Ø
C                                                                       98ØØ
C  ROOT COMPUTES A ROOT OF THE NONLINEAR EQUATION F(X)=Ø                981Ø
C  WHERE F(X) IS A CONTINOUS REAL FUNCTION OF A SINGLE REAL             982Ø
C  VARIABLE X.  THE METHOD USED IS A COMBINATION OF BISECTION           983Ø
C  AND THE SECANT RULE.                                                 984Ø
C                                                                       985Ø
C  NORMAL INPUT CONSISTS OF A CONTINUOS FUNCTION F AND AN               986Ø
C  INTERVAL (B,C) SUCH THAT F(B)*F(C).LE.Ø.Ø.  EACH ITERATION           987Ø
C  FINDS NEW VALUES OF B AND C SUCH THAT THE INTERVAL(B,C) IS           988Ø
C  SHRUNK AND F(B)*F(C).LE.Ø.Ø.  THE STOPPING CRITERION IS             989Ø
C                                                                       99ØØ
C          ABS(B-C).LE.2.Ø*(RELERR*ABS(B)+ABSERR)                       991Ø
C                                                                       992Ø
C  WHERE RELERR=RELATIVE ERROR AND ABSERR=ABSOLUTE ERROR ARE            993Ø
C  INPUT QUANTITIES.  SET THE FLAG, IFLAG, POSITIVE TO INITIALIZE       994Ø
C  THE COMPUTATION.  AS B,C AND IFLAG ARE USED FOR BOTH INPUT AND       995Ø
C  OUTPUT, THEY MUST BE VARIABLES IN THE CALLING PROGRAM.               996Ø
C                                                                       997Ø
C  IF Ø IS A POSSIBLE ROOT, ONE SHOULD NOT CHOOSE ABSERR=Ø.Ø.           998Ø
C                                                                       999Ø
C  THE OUTPUT VALUE OF B IS THE BETTER APPROXIMATION TO A ROOT          1ØØØØ
C  AS B AND C ARE ALWAYS REDEFINED SO THAT ABS(F(B)).LE.ABS(F(C)).      1ØØ1Ø
C                                                                       1ØØ2Ø
C  TO SOLVE THE EQUATION, ROOT MUST EVALUATE F(X) REPEATEDLY. THIS      1ØØ3Ø
C  IS DONE IN THE CALLING PROGRAM.  WHEN AN EVALUATION OF F IS          1ØØ4Ø
C  NEEDED AT T, ROOT RETURNS WITH IFLAG NEGATIVE.  EVALUATE FT=F(T)     1ØØ5Ø
C  AND CALL ROOT AGAIN.  DO NOT ALTER IFLAG.                            1ØØ6Ø
C                                                                       1ØØ7Ø
C  WHEN THE COMPUTATION IS COMPLETE, ROOT RETURNS TO THE CALLING        1ØØ8Ø
C  PROGRAM WITH IFLAG POSITIVE=                                         1ØØ9Ø
C                                                                       1Ø1ØØ
C     IFLAG=1  IF F(B)*F(C).LT.Ø AND THE STOPPING CRITERION IS MET.     1Ø11Ø
C                                                                       1Ø12Ø
C          =2  IF A VALUE B IS FOUND SUCH THAT THE COMPUTED VALUE       1Ø13Ø
C              F(B) IS EXACTLY ZERO.  THE INTERVAL (B,C) MAY NOT        1Ø14Ø
```

```
C              SATISFY THE STOPPING CRITERION.                          1Ø15Ø
C                                                                       1Ø16Ø
C          =3  IF ABS(F(B)) EXCEEDS THE INPUT VALUES ABS(F(B)),         1Ø17Ø
C              ABS(F(C)).  IN THIS CASE IT IS LIKELY THAT B IS CLOSE    1Ø18Ø
C              TO A POLE OF F.                                          1Ø19Ø
C                                                                       1Ø2ØØ
C          =4  IF NO ODD ORDER ROOT WAS FOUND IN THE INTERVAL.  A       1Ø21Ø
C              LOCAL MINIMUM MAY HAVE BEEN OBTAINED.                    1Ø22Ø
C                                                                       1Ø23Ø
C          =5  IF TOO MANY FUNCTION EVALUATIONS WERE MADE.              1Ø24Ø
C              (AS PROGRAMMED, 5ØØ ARE ALLOWED.)                        1Ø25Ø
C                                                                       1Ø26Ø
C  THIS CODE IS A MODIFICATION OF THE CODE ZEROIN WHICH IS COMPLETELY   1Ø27Ø
C  EXPLAINED AND DOCUMENTED IN THE TEXT,  NUMERICAL COMPUTING=  AN      1Ø28Ø
C  INTRODUCTION  BY L. F. SHAMPINE AND R. C. ALLEN.                     1Ø29Ø
C                                                                       1Ø3ØØ
C***********************************************************************1Ø31Ø
C*  THE ONLY MACHINE DEPENDENT CONSTANT IS BASED ON THE MACHINE UNIT   *1Ø32Ø
C*  ROUNDOFF ERROR  U  WHICH IS THE SMALLEST POSITIVE NUMBER SUCH THAT *1Ø33Ø
C*  1.Ø+U .GT. 1.Ø.  U MUST BE CALCULATED AND INSERTED IN THE          *1Ø34Ø
C*  FOLLOWING DATA STATEMENT BEFORE USING ROOT.  THE ROUTINE MACHIN     *1Ø35Ø
C*  CALCULATES  U.                                                      1Ø36Ø
C*  FOR THE CDC 65ØØ, 66ØØ, 76ØØ --                                   *
C*    DATA U /16414ØØØØØØØØØØØØØØØB/
C*  FOR THE IBM 36Ø/37Ø (SINGLE PRECISION) --                          *
C*    DATA U /Z3C1ØØØØØ/
C*  FOR THE IBM 36Ø/37Ø (DOUBLE PRECISION) --                          *
C*    DATA U /Z341ØØØØØØØØØØØØØØ/
C*  FOR THE UNIVAC 11Ø8 AND HONEYWELL 6ØØØ --                          *
C*    DATA U /2.Ø**(-26)/
C*  FOR THE PDP-11 --                                                  *
C*    DATA U /2.Ø**(-23)/
C***********************************************************************1Ø38Ø
C                                                                       1Ø39Ø
      IF(IFLAG.GE.Ø) GO TO 1ØØ                                          1Ø4ØØ
      IFLAG=IABS(IFLAG)                                                 1Ø41Ø
      GO TO (2ØØ,3ØØ,4ØØ), IFLAG                                        1Ø42Ø
  1ØØ RE=AMAX1(RELERR,U)                                                1Ø43Ø
      AE=AMAX1(ABSERR,Ø.Ø)                                              1Ø44Ø
      IC=Ø                                                              1Ø45Ø
      ACBS=ABS(B-C)                                                     1Ø46Ø
      A=C                                                               1Ø47Ø
      T=A                                                               1Ø48Ø
      IFLAG=-1                                                          1Ø49Ø
      RETURN                                                            1Ø5ØØ
  2ØØ FA=FT                                                             1Ø51Ø
      T=B                                                               1Ø52Ø
      IFLAG=-2                                                          1Ø53Ø
      RETURN                                                            1Ø54Ø
  3ØØ FB=FT                                                             1Ø55Ø
      FC=FA                                                             1Ø56Ø
      KOUNT=2                                                           1Ø57Ø
      FX=AMAX1(ABS(FB),ABS(FC))                                         1Ø58Ø
    1 IF(ABS(FC).GE.ABS(FB))GO TO 2                                     1Ø59Ø
C                                                                       1Ø6ØØ
C  INTERCHANGE B AND C SO YHAT ABS(F(B)).LE.ABS(F(C)).                  1Ø61Ø
C                                                                       1Ø62Ø
      A=B                                                               1Ø63Ø
      FA=FB                                                             1Ø64Ø
      B=C                                                               1Ø65Ø
      FB=FC                                                             1Ø66Ø
      C=A                                                               1Ø67Ø
      FC=FA                                                             1Ø68Ø
    2 CMB=Ø.5*(C-B)                                                     1Ø69Ø
      ACMB=ABS(CMB)                                                     1Ø7ØØ
      TOL=RE*ABS(B)+AE                                                  1Ø71Ø
C                                                                       1Ø72Ø
C  TEST STOPPING CRITERION AND FUNCTION COUNT.                          1Ø73Ø
C                                                                       1Ø74Ø
      IF(ACMB.LE.TOL)GO TO 8                                            1Ø75Ø
      IF(KOUNT.GE.5ØØ)GO TO 12                                          1Ø76Ø
C                                                                       1Ø77Ø
C  CALCULATE NEW ITERATE EXPLICITLY AS B+P/Q                            1Ø78Ø
C  WHERE WE ARRANGE P.GE.Ø.  THE IMPLICIT                               1Ø79Ø
C  FORM IS USED TO PREVENT OVERFLOW.                                    1Ø8ØØ
C                                                                       1Ø81Ø
```

```
      P=(B-A)*FB                                                        1Ø82Ø
      Q=FA-FB                                                           1Ø83Ø
      IF(P.GE.Ø.Ø)GO TO 3                                               1Ø84Ø
      P=-P                                                              1Ø85Ø
      Q=-Q                                                              1Ø86Ø
C                                                                       1Ø87Ø
C  UPDATE A, CHECK IF REDUCTION IN THE SIZE OF BRACKETING               1Ø88Ø
C  INTERVAL IS SATISFACTORY.  IF NOT BISECT UNTIL IT IS.                1Ø89Ø
C                                                                       1Ø9ØØ
    3 A=B                                                               1Ø91Ø
      FA=FB                                                             1Ø92Ø
      IC=IC+1                                                           1Ø93Ø
      IF(IC.LT.4)GO TO 4                                                1Ø94Ø
      IF(8.Ø*ACMB.GE.ACBS)GO TO 6                                       1Ø95Ø
      IC=Ø                                                              1Ø96Ø
      ACBS=ACMB                                                         1Ø97Ø
C                                                                       1Ø98Ø
C  TEST FOR TOO SMALL A CHANGE.                                         1Ø99Ø
C                                                                       11ØØØ
    4 IF(P.GT.ABS(Q)*TOL)GO TO 5                                        11Ø1Ø
C                                                                       11Ø2Ø
C  INCREMENT BY TOLERANCE                                               11Ø3Ø
C                                                                       11Ø4Ø
      B=B+SIGN(TOL,CMB)                                                 11Ø5Ø
      GO TO 7                                                           11Ø6Ø
C                                                                       11Ø7Ø
C  ROOT OUGHT TO BE BETWEEN B AND (C+B)/2                               11Ø8Ø
C                                                                       11Ø9Ø
    5 IF(P.GE.CMB*Q)GO TO 6                                             111ØØ
C                                                                       1111Ø
C  USE SECANT RULE.                                                     1112Ø
C                                                                       1113Ø
      B=B+P/Q                                                           1114Ø
      GO TO 7                                                           1115Ø
C                                                                       1116Ø
C  USE BISECTION.                                                       1117Ø
C                                                                       1118Ø
    6 B=Ø.5*(C+B)                                                       1119Ø
C                                                                       112ØØ
C  HAVE COMPLETED COMPUTATION FOR NEW ITERATE B.                        1121Ø
C                                                                       1122Ø
    7 T=B                                                               1123Ø
      IFLAG=-3                                                          1124Ø
      RETURN                                                            1125Ø
 4ØØ FB=FT                                                              1126Ø
      IF(FB.EQ.Ø.Ø)GO TO 9                                              1127Ø
      KOUNT=KOUNT+1                                                     1128Ø
      IF(SIGN(1.Ø,FB).NE.SIGN(1.Ø,FC))GO TO 1                           1129Ø
      C=A                                                               113ØØ
      FC=FA                                                             1131Ø
      GO TO 1                                                           1132Ø
C                                                                       1133Ø
C FINISHED.  SET IFLAG.                                                 1134Ø
C                                                                       1135Ø
    8 IF(SIGN(1.Ø,FB).EQ.SIGN(1.Ø,FC))GO TO 11                          1136Ø
      IF(ABS(FB).GT.FX)GO TO 1Ø                                         1137Ø
      IFLAG=1                                                           1138Ø
      RETURN                                                            1139Ø
    9 IFLAG=2                                                           114ØØ
      RETURN                                                            1141Ø
   1Ø IFLAG=3                                                           1142Ø
      RETURN                                                            1143Ø
   11 IFLAG=4                                                           1144Ø
      RETURN                                                            1145Ø
   12 IFLAG=5                                                           1146Ø
      RETURN                                                            1147Ø
      END                                                               1148Ø
```

ALGORITHM 556
Exponential Integrals [S13]

DONALD E. AMOS
Sandia National Laboratories

Key Words and Phrases: exponential integrals, Miller algorithm, confluent hypergeometric functions
CR Categories: 5.12
Language: Fortran

DESCRIPTION

The Fortran subroutine EXPINT given here is an implementation of [1]. EXPINT has four machine-dependent parameters XCUT, XLIM, ETOL, and EULER which are set into DATA statements. XCUT is a breakpoint such that for $x \leq$ XCUT, the series is evaluated and for $x >$ XCUT, the Miller algorithm is applied. XLIM is the approximate underflow limit for e^{-x}, $x \geq 0$, and ETOL is nominally set to 1.E$-D$ where D is the number of base 10 digits in a word. EULER is the negative Euler constant, $-\gamma$. Thus

```
DATA XCUT, XLIM, ETOL/2.0, 667.0, 1.E-12/
```

would be appropriate for CDC single-precision arithmetic, while

```
DATA XCUT, XLIM, ETOL/1.0, 172.0, 1.E-6/
```

would be appropriate for IBM single-precision arithmetic. The two choices for XCUT reflect the fact that there is a loss of up to two digits on $1 < x \leq 2$ with the series evaluation. This loss can be tolerated on longer word length machines, but not on shorter word length machines.

Maximum accuracy can always be achieved with XCUT = 1. However, for longer word lengths where $D = 14$, the reduction in computation by moving XCUT from 1 to 2, achieved at the expense of two digits of accuracy, seems to be a worthwhile trade-off, and $D = 12$ reflects this modification for CDC machines. $D = 12$ is also more consistent with the accuracy attainable from EXP($-X$) near the underflow limit $X = 667$.

While the subroutine EXPINT is almost portable, the function DIGAM, which computes the psi function at integer arguments, is supplied as a CDC 6600-7600 Fortran function. Modifications for other machines can be made easily from eq. (7) of [1].

The convergence of eq. (2) in [1] is so rapid that the $m = n - 1$ term, which requires $\psi(n)$, is reached for only small values of n. A table of 100 values suffices for virtually all single-precision implementations of EXPINT. An initialization step to generate a higher precision table might be appropriate if multiple precision is anticipated. For CDC single precision or IBM double precision, 36 values suffice for n as high as 10^{12} and relative errors of 10^{-14}

Received August 1978; revised 4 March 1980.
This work was supported by the U.S. Department of Energy under Contract AT(29-1)-789. The U.S. Government's right to retain a nonexclusive royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes is acknowledged.
Author's address: Numerical Mathematics Division, Sandia Laboratories, Albuquerque, NM 87185.

Testing of EXPINT

Extensive testing of EXPINT against a quadrature

$$E_n(x) = e^{-x}x^{n-1}\int_x^\infty \frac{e^{-(t-x)}}{t^n}\,dt$$

was used to check out the routine on parameter ranges

$$E_n(x) = e^{-x}x^{n-1}\int_x^\infty \frac{e^{-(t-x)}}{t^n}\,dt$$

was used to check out the routine on parameter ranges

$$10^{-11} \le \text{TOL} \le 0.1$$
$$0 \le X \le 100$$
$$1 \le N \le 100$$
$$1 \le M \le 100$$
$$\text{XCUT} = 1, 2$$
$$\text{KODE} = 1, 2$$

where KODE is used to scale $E_{N+k}(x)$, $k = 0, 1, \ldots, M-1$ by e^x, if desired. Some spot checks for x and N in the hundreds, as well as special values XCUT $\pm 10^{-13}$, were also made at tolerances TOL as low as 10^{-9}. The quadrature could be relied upon to give the requested accuracy down to TOL = 10^{-11} over the full exponent range of the CDC 6600. For large x, argument reduction in computing e^{-x} will result in decreased accuracy, up to three digits near the underflow limit $x \sim 667$. Hence tolerances smaller than 10^{-11} would require double-precision exponentials to obtain checks for large x.

Relative errors in long sequences, up to 100 terms, were checked for degradation of the recurrence by comparing members of a sequence with single evaluations ($M = 1$) at high accuracies. When $M = 1$, EXPINT uses one of the two basic methods for evaluation rather than recursion.

Relative errors in all of these tests were less than the relative error specified in EXPINT to generate the test values.

REFERENCE

1. Amos, D.E. Computation of exponential integrals. *ACM Trans. Math. Softw. 6,* 3 (Sept. 1980), 365-377.

ALGORITHM

```
      SUBROUTINE EXPINT(X, N, KODE, M, TOL, EN, IERR)                    EXP   1Ø
C                                                                        EXP   2Ø
C     WRITTEN BY D.E. AMOS, SANDIA LABORATORIES, ALBUQUERQUE, NM, 87185  EXP   3Ø
C                                                                        EXP   4Ø
C     REFERENCE                                                          EXP   5Ø
C         COMPUTATION OF EXPONENTIAL INTEGRALS BY D.E. AMOS, ACM         EXP   6Ø
C         TRANS. MATH SOFTWARE, 198Ø                                     EXP   7Ø
C                                                                        EXP   8Ø
C     ABSTRACT                                                           EXP   9Ø
C         EXPINT COMPUTES M MEMBER SEQUENCES OF EXPONENTIAL INTEGRALS    EXP  1ØØ
C         E(N+K,X), K=Ø,1,...,M-1 FOR N.GE.1 AND X.GE.Ø.  THE POWER      EXP  11Ø
C         SERIES IS IMPLEMENTED FOR X.LE.XCUT AND THE CONFLUENT          EXP  12Ø
C         HYPERGEOMETRIC REPRESENTATION                                  EXP  13Ø
C                                                                        EXP  14Ø
C                     E(A,X) = EXP(-X)*(X**(A-1))*U(A,A,X)               EXP  15Ø
C                                                                        EXP  16Ø
C         IS COMPUTED FOR X.GT.XCUT. SINCE SEQUENCES ARE COMPUTED IN A   EXP  17Ø
C         STABLE FASHION BY RECURRING AWAY FROM X, A IS SELECTED AS THE  EXP  18Ø
C         INTEGER CLOSEST TO X WITHIN THE CONSTRAINT N.LE.A.LE.N+M-1.    EXP  19Ø
C         FOR THE U COMPUTATION  A IS FURTHER MODIFIED TO BE THE         EXP  2ØØ
C         NEAREST EVEN INTEGER. INDICES ARE CARRIED FORWARD OR           EXP  21Ø
C         BACKWARD BY THE TWO TERM RECURSION RELATION                    EXP  22Ø
C                                                                        EXP  23Ø
C                     K*E(K+1,X) + X*E(K,X) = EXP(-X)                    EXP  24Ø
```

```
C                                                                       EXP  25Ø
C         ONCE E(A,X) IS COMPUTED. THE U FUNCTION IS COMPUTED BY MEANS  EXP  26Ø
C         OF THE BACKWARD RECURSIVE MILLER ALGORITHM APPLIED TO THE     EXP  27Ø
C         THREE TERM CONTIGUOUS RELATION FOR U(A+K,A,X), K=Ø,1,...      EXP  28Ø
C         THIS PRODUCES ACCURATE RATIOS AND DETERMINES U(A+K,A,X),AND   EXP  29Ø
C         HENCE E(A,X), TO WITHIN A MULTIPLICATIVE CONSTANT C.          EXP  3ØØ
C         ANOTHER CONTIGUOUS RELATION APPLIED TO C*U(A,A,X) AND         EXP  31Ø
C         C*U(A+1,A,X) GETS C*U(A+1,A+1,X), A QUANTITY PROPORTIONAL TO  EXP  32Ø
C         E(A+1,X). THE NORMALIZING CONSTANT C IS OBTAINED FROM THE     EXP  33Ø
C         TWO TERM RECURSION RELATION ABOVE WITH K=A.                   EXP  34Ø
C                                                                       EXP  35Ø
C         MACHINE DEPENDENT PARAMETERS - XCUT, XLIM, ETOL, EULER, DIGAM EXP  36Ø
C                                                                       EXP  37Ø
C         EXPINT WRITES ERROR DIAGNOSTICS TO LOGICAL UNIT 3             EXP  38Ø
C                                                                       EXP  39Ø
C     DESCRIPTION OF ARGUMENTS                                          EXP  4ØØ
C                                                                       EXP  41Ø
C         INPUT                                                         EXP  42Ø
C           X       X.GT.Ø.Ø FOR N=1 AND  X.GE.Ø.Ø FOR N.GE.2           EXP  43Ø
C           N       ORDER OF THE FIRST MEMBER OF THE SEQUENCE, N.GE.1   EXP  44Ø
C           KODE    A SELECTION PARAMETER FOR SCALED VALUES             EXP  45Ø
C                   KODE=1   RETURNS        E(N+K,X), K=Ø,1,...,M-1.    EXP  46Ø
C                       =2   RETURNS EXP(X)*E(N+K,X), K=Ø,1,...,M-1.    EXP  47Ø
C           M       NUMBER OF EXPONENTIAL INTEGRALS IN THE SEQUENCE,    EXP  48Ø
C                   M.GE.1                                              EXP  49Ø
C           TOL     RELATIVE ACCURACY WANTED, ETOL.LE.TOL.LE.Ø.1        EXP  5ØØ
C                   ETOL=1.E-12                                         EXP  51Ø
C                                                                       EXP  52Ø
C         OUTPUT                                                        EXP  53Ø
C           EN      A VECTOR OF DIMENSION AT LEAST M CONTAINING VALUES  EXP  54Ø
C                   EN(K) = E(N+K-1,X) OR EXP(X)*E(N+K-1,X), K=1,M      EXP  55Ø
C                   DEPENDING ON KODE                                   EXP  56Ø
C           IERR    UNDERFLOW INDICATOR                                 EXP  57Ø
C                   IERR=Ø   A NORMAL RETURN                            EXP  58Ø
C                       =1   X EXCEEDS XLIM AND AN UNDERFLOW OCCURS.    EXP  59Ø
C                            EN(K)=Ø.Ø , K=1,M RETURNED ON KODE=1       EXP  6ØØ
C                            XLIM=667.                                  EXP  61Ø
C                                                                       EXP  62Ø
C     ERROR CONDITIONS                                                  EXP  63Ø
C         AN IMPROPER INPUT PARAMETER IS A FATAL ERROR                  EXP  64Ø
C         UNDERFLOW IS A NON FATAL ERROR. ZERO ANSWERS ARE RETURNED.    EXP  65Ø
C                                                                       EXP  66Ø
      DIMENSION EN(1), A(99), B(99), Y(2)                               EXP  67Ø
C                                                                       EXP  68Ø
      DATA XCUT, XLIM, ETOL /2.ØEØ,667.ØEØ,1.ØE-12/                     EXP  69Ø
      DATA EULER /-5.772156649Ø1533E-Ø1/                                EXP  7ØØ
      DATA LUN /3/                                                      EXP  71Ø
C                                                                       EXP  72Ø
      IF (N.LT.1) GO TO 26Ø                                             EXP  73Ø
      IF (KODE.LT.1 .OR. KODE.GT.2) GO TO 27Ø                           EXP  74Ø
      IF (M.LT.1) GO TO 28Ø                                             EXP  75Ø
      IF (TOL.LT.ETOL .OR. TOL.GT.Ø.1EØ) GO TO 29Ø                      EXP  76Ø
C                                                                       EXP  77Ø
      IERR = Ø                                                          EXP  78Ø
      IF (X.GT.XCUT) GO TO 1ØØ                                          EXP  79Ø
      IF (X.LT.Ø.ØEØ) GO TO 3ØØ                                         EXP  8ØØ
      IF (X.EQ.Ø.ØEØ .AND. N.EQ.1) GO TO 31Ø                            EXP  81Ø
      IF (X.EQ.Ø.ØEØ .AND. N.GT.1) GO TO 8Ø                             EXP  82Ø
C                                                                       EXP  83Ø
C     SERIES FOR E(N,X) FOR X.LE.XCUT                                   EXP  84Ø
C                                                                       EXP  85Ø
      IX = INT(X+Ø.5EØ)                                                 EXP  86Ø
C     ICASE=1 MEANS INTEGER CLOSEST TO X IS 2 AND N=1                   EXP  87Ø
C     ICASE=2 MEANS INTEGER CLOSEST TO X IS Ø,1, OR 2 AND N.GE.2        EXP  88Ø
      ICASE = 2                                                         EXP  89Ø
      IF (IX.GT.N) ICASE = 1                                            EXP  9ØØ
      NM = N - ICASE + 1                                                EXP  91Ø
      ND = NM + 1                                                       EXP  92Ø
      IND = 3 - ICASE                                                   EXP  93Ø
      MU = M - IND                                                      EXP  94Ø
      ML = 1                                                            EXP  95Ø
      KS = ND                                                           EXP  96Ø
      FNM = FLOAT(NM)                                                   EXP  97Ø
      S = Ø.ØEØ                                                         EXP  98Ø
      XTOL = 3.ØEØ*TOL                                                  EXP  99Ø
      IF (ND.EQ.1) GO TO 1Ø                                             EXP 1ØØØ
      XTOL = Ø.3333EØ*TOL                                               EXP 1Ø1Ø
```

```
      S = 1.ØEØ/FNM                                                    EXP 1Ø2Ø
   1Ø CONTINUE                                                         EXP 1Ø3Ø
      AA = 1.ØEØ                                                       EXP 1Ø4Ø
      AK = 1.ØEØ                                                       EXP 1Ø5Ø
      DO 5Ø I=1,35                                                     EXP 1Ø6Ø
        AA = -AA*X/AK                                                  EXP 1Ø7Ø
        IF (I.EQ.NM) GO TO 3Ø                                          EXP 1Ø8Ø
        S = S - AA/(AK-FNM)                                            EXP 1Ø9Ø
        IF (ABS(AA).LE.XTOL*ABS(S)) GO TO 2Ø                           EXP 11ØØ
        AK = AK + 1.ØEØ                                                EXP 111Ø
        GO TO 5Ø                                                       EXP 112Ø
   2Ø   CONTINUE                                                       EXP 113Ø
        IF (I.LT.2) GO TO 4Ø                                           EXP 114Ø
        IF (ND-2.GT.I .OR. I.GT.ND-1) GO TO 6Ø                         EXP 115Ø
        AK = AK + 1.ØEØ                                                EXP 116Ø
        GO TO 5Ø                                                       EXP 117Ø
   3Ø   S = S + AA*(-ALOG(X)+DIGAM(ND))                                EXP 118Ø
        XTOL = 3.ØEØ*TOL                                               EXP 119Ø
   4Ø   AK = AK + 1.ØEØ                                                EXP 12ØØ
   5Ø CONTINUE                                                         EXP 121Ø
      GO TO 32Ø                                                        EXP 122Ø
   6Ø IF (ND.EQ.1) S = S + (-ALOG(X)+EULER)                            EXP 123Ø
      IF (KODE.EQ.2) S = S*EXP(X)                                      EXP 124Ø
      EN(1) = S                                                        EXP 125Ø
      EMX = 1.ØEØ                                                      EXP 126Ø
      IF (M.EQ.1) GO TO 7Ø                                             EXP 127Ø
      EN(IND) = S                                                      EXP 128Ø
      AA = FLOAT(KS)                                                   EXP 129Ø
      IF (KODE.EQ.1) EMX = EXP(-X)                                     EXP 13ØØ
      GO TO (22Ø, 24Ø), ICASE                                          EXP 131Ø
   7Ø IF (ICASE.EQ.2) RETURN                                           EXP 132Ø
      IF (KODE.EQ.1) EMX = EXP(-X)                                     EXP 133Ø
      EN(1) = (EMX-S)/X                                                EXP 134Ø
      RETURN                                                           EXP 135Ø
   8Ø CONTINUE                                                         EXP 136Ø
      DO 9Ø I=1,M                                                      EXP 137Ø
        EN(I) = 1.ØEØ/FLOAT(N+I-2)                                     EXP 138Ø
   9Ø CONTINUE                                                         EXP 139Ø
      RETURN                                                           EXP 14ØØ
C                                                                      EXP 141Ø
C     BACKWARD RECURSIVE MILLER ALGORITHM FOR                          EXP 142Ø
C             E(N,X)=EXP(-X)*(X**(N-1))*U(N,N,X)                       EXP 143Ø
C     WITH RECURSION AWAY FROM N=INTEGER CLOSEST TO X.                 EXP 144Ø
C     U(A,B,X) IS THE SECOND CONFLUENT HYPERGEOMETRIC FUNCTION         EXP 145Ø
C                                                                      EXP 146Ø
  1ØØ CONTINUE                                                         EXP 147Ø
      EMX = 1.ØEØ                                                      EXP 148Ø
      IF (KODE.EQ.2) GO TO 13Ø                                         EXP 149Ø
      IF (X.LE.XLIM) GO TO 12Ø                                         EXP 15ØØ
      IERR = 1                                                         EXP 151Ø
      DO 11Ø I=1,M                                                     EXP 152Ø
        EN(I) = Ø.ØEØ                                                  EXP 153Ø
  11Ø CONTINUE                                                         EXP 154Ø
      RETURN                                                           EXP 155Ø
  12Ø EMX = EXP(-X)                                                    EXP 156Ø
  13Ø CONTINUE                                                         EXP 157Ø
      IX = INT(X+Ø.5EØ)                                                EXP 158Ø
      KN = N + M - 1                                                   EXP 159Ø
      IF (KN.LE.IX) GO TO 14Ø                                          EXP 16ØØ
      IF (N.LT.IX .AND. IX.LT.KN) GO TO 17Ø                            EXP 161Ø
      IF (N.GE.IX) GO TO 16Ø                                           EXP 162Ø
      GO TO 34Ø                                                        EXP 163Ø
  14Ø ICASE = 1                                                        EXP 164Ø
      KS = KN                                                          EXP 165Ø
      ML = M - 1                                                       EXP 166Ø
      MU = -1                                                          EXP 167Ø
      IND = M                                                          EXP 168Ø
      IF (KN.GT.1) GO TO 18Ø                                           EXP 169Ø
  15Ø KS = 2                                                           EXP 17ØØ
      ICASE = 3                                                        EXP 171Ø
      GO TO 18Ø                                                        EXP 172Ø
  16Ø ICASE = 2                                                        EXP 173Ø
      IND = 1                                                          EXP 174Ø
      KS = N                                                           EXP 175Ø
      MU = M - 1                                                       EXP 176Ø
      IF (N.GT.1) GO TO 18Ø                                            EXP 177Ø
```

```
      IF (KN.EQ.1) GO TO 15Ø                                          EXP 178Ø
      IX = 2                                                          EXP 179Ø
  17Ø ICASE = 1                                                       EXP 18ØØ
      KS = IX                                                         EXP 181Ø
      ML = IX - N                                                     EXP 182Ø
      IND = ML + 1                                                    EXP 183Ø
      MU = KN - IX                                                    EXP 184Ø
  18Ø CONTINUE                                                        EXP 185Ø
      IK = KS/2                                                       EXP 186Ø
      AH = FLOAT(IK)                                                  EXP 187Ø
      JSET = 1 + KS - (IK+IK)                                         EXP 188Ø
C     START COMPUTATION FOR                                           EXP 189Ø
C              EN(IND) = C*U( A , A ,X)     JSET=1                    EXP 19ØØ
C              EN(IND) = C*U(A+1,A+1,X)     JSET=2                    EXP 191Ø
C     FOR AN EVEN INTEGER A.                                          EXP 192Ø
      IC = Ø                                                          EXP 193Ø
      AA = AH + AH                                                    EXP 194Ø
      AAMS = AA - 1.ØEØ                                               EXP 195Ø
      AAMS = AAMS*AAMS                                                EXP 196Ø
      TX = X + X                                                      EXP 197Ø
      FX = TX + TX                                                    EXP 198Ø
      AK = AH                                                         EXP 199Ø
      XTOL = TOL                                                      EXP 2ØØØ
      IF (TOL.LE.1.ØE-3) XTOL = 2Ø.ØEØ*TOL                            EXP 2Ø1Ø
      CT = AAMS + FX*AH                                               EXP 2Ø2Ø
      EM = (AH+1.ØEØ)/((X+AA)*XTOL*SQRT(CT))                          EXP 2Ø3Ø
      BK = AA                                                         EXP 2Ø4Ø
      CC = AH*AH                                                      EXP 2Ø5Ø
C     FORWARD RECURSION FOR P(IC),P(IC+1) AND INDEX IC FOR BACKWARD   EXP 2Ø6Ø
C     RECURSION                                                       EXP 2Ø7Ø
      P1 = Ø.ØEØ                                                      EXP 2Ø8Ø
      P2 = 1.ØEØ                                                      EXP 2Ø9Ø
  19Ø CONTINUE                                                        EXP 21ØØ
      IF (IC.EQ.99) GO TO 33Ø                                         EXP 211Ø
      IC = IC + 1                                                     EXP 212Ø
      AK = AK + 1.ØEØ                                                 EXP 213Ø
      AT = BK/(BK+AK+CC+FLOAT(IC))                                    EXP 214Ø
      BK = BK + AK + AK                                               EXP 215Ø
      A(IC) = AT                                                      EXP 216Ø
      BT = (AK+AK+X)/(AK+1.ØEØ)                                       EXP 217Ø
      B(IC) = BT                                                      EXP 218Ø
      PT = P2                                                         EXP 219Ø
      P2 = BT*P2 - AT*P1                                              EXP 22ØØ
      P1 = PT                                                         EXP 221Ø
      CT = CT + FX                                                    EXP 222Ø
      EM = EM*AT*(1.ØEØ-TX/CT)                                        EXP 223Ø
      IF (EM*(AK+1.ØEØ).GT.P1*P1) GO TO 19Ø                           EXP 224Ø
      ICT = IC                                                        EXP 225Ø
      KK = IC + 1                                                     EXP 226Ø
      BT = TX/(CT+FX)                                                 EXP 227Ø
      Y2 = (BK/(BK+CC+FLOAT(KK)))*(P1/P2)*(1.ØEØ-BT+Ø.375EØ*BT*BT)    EXP 228Ø
      Y1 = 1.ØEØ                                                      EXP 229Ø
C     BACKWARD RECURRENCE FOR                                         EXP 23ØØ
C              Y1=             C*U( A ,A,X)                         EXP 231Ø
C              Y2= C*(A/(1+A/2))*U(A+1,A,X)                         EXP 232Ø
      DO 2ØØ K=1,ICT                                                  EXP 233Ø
        KK = KK - 1                                                   EXP 234Ø
        YT = Y1                                                       EXP 235Ø
        Y1 = (B(KK)*Y1-Y2)/A(KK)                                      EXP 236Ø
        Y2 = YT                                                       EXP 237Ø
  2ØØ CONTINUE                                                        EXP 238Ø
C     THE CONTIGUOUS RELATION                                         EXP 239Ø
C              X*U(B,C+1,X)=(C-B)*U(B,C,X)+U(B-1,C,X)                 EXP 24ØØ
C     WITH  B=A+1 , C=A IS USED FOR                                   EXP 241Ø
C              Y(2) = C * U(A+1,A+1,X)                              EXP 242Ø
C     X IS INCORPORATED INTO THE NORMALIZING RELATION FOR CNORM.      EXP 243Ø
      Y(1) = Y1                                                       EXP 244Ø
      Y(2) = Y1 - Y2*(AH+1.ØEØ)/AA                                    EXP 245Ø
      CNORM = EMX/(AA*Y(2)+X*Y(1))                                    EXP 246Ø
      IF (ICASE.EQ.3) GO TO 21Ø                                       EXP 247Ø
      EN(IND) = CNORM*Y(JSET)                                         EXP 248Ø
      IF (M.EQ.1) RETURN                                              EXP 249Ø
      AA = FLOAT(KS)                                                  EXP 25ØØ
      GO TO (22Ø, 24Ø), ICASE                                         EXP 251Ø
C                                                                     EXP 252Ø
C     RECURSION SECTION  N*E(N+1,X) + X*E(N,X)=EMX                    EXP 253Ø
C                                                                     EXP 254Ø
```

```
  21Ø EN(1) = (EMX-CNORM*Y(1))/X                                        EXP 255Ø
      RETURN                                                            EXP 256Ø
  22Ø K = IND - 1                                                       EXP 257Ø
      DO 23Ø I=1,ML                                                     EXP 258Ø
        AA = AA - 1.ØEØ                                                 EXP 259Ø
        EN(K) = (EMX-AA*EN(K+1))/X                                      EXP 26ØØ
        K = K - 1                                                       EXP 261Ø
  23Ø CONTINUE                                                          EXP 262Ø
      IF (MU.LE.Ø) RETURN                                               EXP 263Ø
      AA = FLOAT(KS)                                                    EXP 264
  24Ø K = IND                                                           EXP 265Ø
      DO 25Ø I=1,MU                                                     EXP 266Ø
        EN(K+1) = (EMX-X*EN(K))/AA                                      EXP 267Ø
        AA = AA + 1.ØEØ                                                 EXP 268Ø
        K = K + 1                                                       EXP 269Ø
  25Ø CONTINUE                                                          EXP 27ØØ
      RETURN                                                            EXP 271Ø
C                                                                       EXP 272Ø
C                                                                       EXP 273Ø
  26Ø WRITE (LUN,99999)                                                 EXP 274Ø
      RETURN                                                            EXP 275Ø
  27Ø WRITE (LUN,99998)                                                 EXP 276Ø
      RETURN                                                            EXP 277Ø
  28Ø WRITE (LUN,99997)                                                 EXP 278Ø
      RETURN                                                            EXP 279Ø
  29Ø WRITE (LUN,99996)                                                 EXP 28ØØ
      RETURN                                                            EXP 281Ø
  3ØØ WRITE (LUN,99995)                                                 EXP 282Ø
      RETURN                                                            EXP 283Ø
  31Ø WRITE (LUN,99994)                                                 EXP 284Ø
      RETURN                                                            EXP 285Ø
  32Ø WRITE (LUN,99993)                                                 EXP 286Ø
      RETURN                                                            EXP 287Ø
  33Ø WRITE (LUN,99992)                                                 EXP 288Ø
      RETURN                                                            EXP 289Ø
  34Ø WRITE (LUN,99991)                                                 EXP 29ØØ
      RETURN                                                            EXP 291Ø
99999 FORMAT (32H IN EXPINT, N NOT GREATER THAN Ø)                       EXP 292Ø
99998 FORMAT (27H IN EXPINT, KODE NOT 1 OR 2)                           EXP 293Ø
99997 FORMAT (32H IN EXPINT, M NOT GREATER THAN Ø)                      EXP 294Ø
99996 FORMAT (33H IN EXPINT, TOL NOT WITHIN LIMITS)                     EXP 295Ø
99995 FORMAT (37H IN EXPINT, X IS NOT ZERO OR POSITIVE)                 EXP 296Ø
99994 FORMAT (46H IN EXPINT, THE EXPONENTIAL INTEGRAL IS NOT DE,        EXP 297Ø
     * 21HFINED FOR X=Ø AND N=1)                                        EXP 298Ø
99993 FORMAT (46H IN EXPINT, RELATIVE ERROR TEST FOR SERIES TER,        EXP 299Ø
     * 28HMINATION NOT MET IN 36 TERMS)                                 EXP 3ØØØ
99992 FORMAT (46H IN EXPINT, TERMINATION TEST FOR MILLER ALGORI,        EXP 3Ø1Ø
     * 23HTHM NOT MET IN 99 STEPS)                                      EXP 3Ø2Ø
99991 FORMAT (46H IN EXPINT, AN ERROR IN PLACING INT(X+Ø.5) WIT,        EXP 3Ø3Ø
     * 47HH RESPECT TO N AND N+M-1 OCCURRED FOR X.GT.XCUT)              EXP 3Ø4Ø
      END                                                               EXP 3Ø5Ø


      FUNCTION DIGAM(N)                                                 DIG   1Ø
C                                                                       DIG   2Ø
C     THIS SUBROUTINE RETURNS VALUES OF PSI(X)=DERIVATIVE OF LOG        DIG   3Ø
C     GAMMA(X), X.GT.Ø.Ø AT INTEGER ARGUMENTS. A TABLE LOOK-UP IS       DIG   4Ø
C     PERFORMED FOR N.LE.1ØØ, AND THE ASYMPTOTIC EXPANSION IS           DIG   5Ø
C     EVALUATED FOR N.GT.1ØØ.                                           DIG   6Ø
C                                                                       DIG   7Ø
      DIMENSION B(4), C(1ØØ), C1(32), C2(27), C3(22), C4(19)            DIG   8Ø
      EQUIVALENCE (C(1),C1(1))                                          DIG   9Ø
      EQUIVALENCE (C(33),C2(1))                                         DIG  1ØØ
      EQUIVALENCE (C(6Ø),C3(1))                                         DIG  11Ø
      EQUIVALENCE (C(82),C4(1))                                         DIG  12Ø
C                                                                       DIG  13Ø
      DATA C1 /-5.772156649Ø153E-Ø1,4.22784335Ø98467E-Ø1,               DIG  14Ø
     * 9.22784335Ø98467E-Ø1,1.2561176684318ØE+ØØ,1.5Ø611766843188E+ØØ,  DIG  15Ø
     * 1.7Ø611766843188E+ØØ,1.8727843350984 7E+ØØ,2.Ø1564147795561E+ØØ,  DIG  16Ø
     * 2.14Ø64147795561E+ØØ,2.251752589Ø6672E+ØØ,2.351752589Ø6672E+ØØ,  DIG  17Ø
     * 2.44266167997581E+ØØ,2.525995Ø133Ø915E+ØØ,2.6Ø2918Ø9Ø23222E+ØØ,  DIG  18Ø
     * 2.67434666166Ø79E+ØØ,2.741Ø1332832746E+ØØ,2.8Ø351332832746E+ØØ,  DIG  19Ø
     * 2.86233685773923E+ØØ,2.91789241329478E+ØØ,2.97Ø52399224215E+ØØ,  DIG  2ØØ
     * 3.Ø2Ø52399224215E+ØØ,3.Ø6814Ø398612ØE+ØØ,3.11359758531574E+ØØ,  DIG  21Ø
     * 3.157Ø7584618531E+ØØ,3.19874251285197E+ØØ,3.23874251285197E+ØØ,  DIG  22Ø
     * 3.2772Ø4Ø5131351E+ØØ,3.3142410883505 5E+ØØ,3.3499553745Ø6484E+ØØ,  DIG  23Ø
```

```
     * 3.38443813268552E+ØØ,3.417771466Ø1886E+ØØ,3.45ØØ2953Ø53499E+ØØ/  DIG  24Ø
      DATA C2 /3.48127953Ø53499E+ØØ,3.51158256Ø838Ø2E+ØØ,                DIG  25Ø
     * 3.54Ø9943255439ØE+ØØ,3.5695657541153 3E+ØØ,3.59734353189311E+ØØ,  DIG  26Ø
     * 3.62437Ø55892Ø13E+ØØ,3.65Ø68634839382E+ØØ,3.676327374Ø3484E+ØØ,  DIG  27Ø
     * 3.7Ø1327374Ø3484E+ØØ,3.72571761793728E+ØØ,3.74952714174681E+ØØ,  DIG  28Ø
     * 3.7727829557ØØ29E+ØØ,3.79551Ø22842757E+ØØ,3.81773245Ø64979E+ØØ,  DIG  29Ø
     * 3.8394715810 8457E+ØØ,3.86Ø74817682925E+ØØ,3.88158151Ø16259E+ØØ,  DIG  3ØØ
     * 3.9Ø198967342789E+ØØ,3.92198967342789E+ØØ,3.94159751656515E+ØØ,  DIG  31Ø
     * 3.96Ø82828579592E+ØØ,3.97969621Ø32422E+ØØ,3.99821472884274E+ØØ,  DIG  32Ø
     * 4.Ø16396547Ø2455E+ØØ,4.Ø342536898817ØE+ØØ,4.Ø517975495 3Ø82E+ØØ,  DIG  33Ø
     * 4.Ø69Ø3892884117E+ØØ/                                            DIG  34Ø
      DATA C3 /4.Ø85988Ø8138354E+ØØ,4.1Ø2654748Ø5Ø2ØE+ØØ,               DIG  35Ø
     * 4.119Ø4819Ø67316E+ØØ,4.13517722293122E+ØØ,4.151Ø5Ø2388Ø424E+ØØ,  DIG  36Ø
     * 4.1666752388Ø424E+ØØ,4.182Ø5985418885E+ØØ,4.19721136934Ø37E+ØØ,  DIG  37Ø
     * 4.21213674247 47ØE+ØØ,4.22684262482764E+ØØ,4.241335378 45Ø82E+ØØ,  DIG  38Ø
     * 4.255621Ø9273654E+ØØ,4.2697Ø559977879E+ØØ,4.28359448866768E+ØØ,  DIG  39Ø
     * 4.29729311 88Ø467E+ØØ,4.31Ø8Ø663231818E+ØØ,4.32413996565151E+ØØ,  DIG  4ØØ
     * 4.33729786Ø38836E+ØØ,4.35Ø28487337537E+ØØ,4.3631Ø538619588E+ØØ,  DIG  41Ø
     * 4.37576361Ø4398E+ØØ,4.38826361Ø4398E+ØØ/                         DIG  42Ø
      DATA C4 /4.4ØØ6Ø9293Ø5633E+ØØ,4.4128Ø4415ØØ755E+ØØ,               DIG  43Ø
     * 4.4248526Ø777863E+ØØ,4.4367573696834ØE+ØØ,4.448522Ø7556575E+ØØ,  DIG  44Ø
     * 4.46Ø14998254249E+ØØ,4.4716442354 16Ø6E+ØØ,4.483ØØ787177969E+ØØ,  DIG  45Ø
     * 4.49424382683587E+ØØ,4.5Ø535493794698E+ØØ,4.51634394893599E+ØØ,  DIG  46Ø
     * 4.52721351415338E+ØØ,4.5379662Ø232543E+ØØ,4.5486Ø45ØØ19777E+ØØ,  DIG  47Ø
     * 4.55913Ø81598724E+ØØ,4.56954748265391E+ØØ,4.579856761ØØ442E+ØØ,  DIG  48Ø
     * 4.59ØØ6Ø8426 37Ø8E+ØØ,4.6ØØ1618527380 9E+ØØ/                     DIG  49Ø
C                                                                       DIG  5ØØ
      DATA B /1.66666666666667E-Ø1,-3.33333333333333E-Ø2,               DIG  51Ø
     * 2.38Ø95238Ø95238E-Ø2,-3.33333333333333E-Ø2/                      DIG  52Ø
C                                                                       DIG  53Ø
      IF (N.GT.1ØØ) GO TO 1Ø                                           DIG  54Ø
      DIGAM = C(N)                                                     DIG  55Ø
      RETURN                                                           DIG  56Ø
   1Ø FN = N                                                           DIG  57Ø
      AX = 1.ØEØ                                                       DIG  58Ø
      AK = 2.ØEØ                                                       DIG  59Ø
      S = -Ø.5EØ/FN                                                    DIG  6ØØ
      IF (FN.GT.1.E+8) GO TO 3Ø                                        DIG  61Ø
      FN2 = FN*FN                                                      DIG  62Ø
      DO 2Ø K=1,3                                                      DIG  63Ø
        AX = AX*FN2                                                    DIG  64Ø
        S = S - B(K)/(AX*AK)                                           DIG  65Ø
        AK = AK + 2.ØEØ                                                DIG  66Ø
   2Ø CONTINUE                                                         DIG  67Ø
   3Ø CONTINUE                                                         DIG  68Ø
      DIGAM = S + ALOG(FN)                                             DIG  69Ø
      RETURN                                                           DIG  7ØØ
      END                                                              DIG  71Ø


C     PROGRAM TSTEXP(INPUT,OUTPUT,TAPE3=OUTPUT)                        ØØØØØØ1Ø
C                                                                      ØØØØØØ2Ø
C     PROGRAM TO TEST SUBROUTINE EXPINT AGAINST AN ADAPTIVE QUADRATURE. ØØØØØØ3Ø
C     PARAMETER VALUES ARE PRINTED AND, IN THE EVENT THAT THE RELATIVE  ØØØØØØ4Ø
C     ERROR TEST IS NOT SATISFIED, X, ERROR, N, AND KODE ARE ALSO       ØØØØØØ5Ø
C     PRINTED. AN OUTPUT WITH ONLY PARAMETER VALUES INDICATES THAT ALL  ØØØØØØ6Ø
C     TESTS WERE PASSED. GAUS8 COMPUTES THE QUADRATURES.                ØØØØØØ7Ø
C                                                                      ØØØØØØ8Ø
      DIMENSION XTOL(1Ø), EN(5Ø), EV(5Ø)                               ØØØØØØ9Ø
      IOUT = 3                                                         ØØØØØ1ØØ
      NL = 1                                                           ØØØØØ11Ø
      NU = 16                                                          ØØØØØ12Ø
      NINC = 5                                                         ØØØØØ13Ø
      ML = 1                                                           ØØØØØ14Ø
      MU = 25                                                          ØØØØØ15Ø
      MINC = 8                                                         ØØØØØ16Ø
      KM = 5                                                           ØØØØØ17Ø
      JL = 1                                                           ØØØØØ18Ø
      JU = 4Ø                                                          ØØØØØ19Ø
      JINC = 3                                                         ØØØØØ2ØØ
      XTOL(1) = 1.ØE-2                                                 ØØØØØ21Ø
      DO 1Ø I=2,3                                                      ØØØØØ22Ø
        XTOL(I) = XTOL(I-1)*1.ØE-3                                     ØØØØØ23Ø
   1Ø CONTINUE                                                         ØØØØØ24Ø
      DO 8Ø IT=1,3                                                     ØØØØØ25Ø
        TOL = XTOL(IT)                                                 ØØØØØ26Ø
```

```
      TOLA = AMAX1(1.ØE-12,TOL/1Ø.ØEØ)                                  ØØØØØ27Ø
      BTOL = TOL                                                        ØØØØØ28Ø
      WRITE (IOUT,99999) TOL                                            ØØØØØ29Ø
      DO 7Ø M=ML,MU,MINC                                                ØØØØØ3ØØ
        WRITE (IOUT,99998) M                                            ØØØØØ31Ø
        DO 6Ø N=NL,NU,NINC                                              ØØØØØ32Ø
          WRITE (IOUT,99997) N                                          ØØØØØ33Ø
          DO 5Ø J=JL,JU,JINC                                            ØØØØØ34Ø
            X = FLOAT(J-1)/5.ØEØ                                        ØØØØØ35Ø
            EX = EXP(-X)                                                ØØØØØ36Ø
            IF (X.EQ.Ø. .AND. N.EQ.1) GO TO 5Ø                          ØØØØØ37Ø
            CALL EXPINT(X, N, 1, M, TOL, EN, IERR)                      ØØØØØ38Ø
            CALL EXPINT(X, N, 2, M, TOL, EV, IERR)                      ØØØØØ39Ø
            DO 4Ø K=1,M,KM                                              ØØØØØ4ØØ
              IF (X.GT.Ø.) GO TO 2Ø                                     ØØØØØ41Ø
              IF (N+K.EQ.2) GO TO 4Ø                                    ØØØØØ42Ø
              Y = 1.ØEØ/FLOAT(N+K-2)                                    ØØØØØ43Ø
              YY = Y                                                    ØØØØØ44Ø
              GO TO 3Ø                                                  ØØØØØ45Ø
   2Ø         CONTINUE                                                  ØØØØØ46Ø
              NN = N + K - 1                                            ØØØØØ47Ø
              YY = EINT(NN,X,TOLA,2)                                    ØØØØØ48Ø
              Y = YY*EX                                                 ØØØØØ49Ø
   3Ø         CONTINUE                                                  ØØØØØ5ØØ
              ER = ABS((Y-EN(K))/Y)                                     ØØØØØ51Ø
              KODE = 1                                                  ØØØØØ52Ø
              IF (ER.GT.BTOL) WRITE (IOUT,99996) X, ER, NN, KODE        ØØØØØ53Ø
              KODE = 2                                                  ØØØØØ54Ø
              ERR = ABS((YY-EV(K))/YY)                                  ØØØØØ55Ø
              IF (ERR.GT.BTOL) WRITE (IOUT,99996) X, ERR, NN, KODE      ØØØØØ56Ø
   4Ø       CONTINUE                                                    ØØØØØ57Ø
   5Ø     CONTINUE                                                      ØØØØØ58Ø
   6Ø   CONTINUE                                                        ØØØØØ59Ø
   7Ø CONTINUE                                                          ØØØØØ6ØØ
   8Ø CONTINUE                                                          ØØØØØ61Ø
      STOP                                                              ØØØØØ62Ø
99999 FORMAT (1HØ, 5H TOL=, E15.4/)                                     ØØØØØ63Ø
99998 FORMAT (1HØ, 2HM=, I5/)                                           ØØØØØ64Ø
99997 FORMAT (3X, 2HN=, I5)                                             ØØØØØ65Ø
99996 FORMAT (2E15.6, 2I5)                                              ØØØØØ66Ø
      END                                                               ØØØØØ67Ø


      FUNCTION EINT(N, X, TOL, KODE)                                    ØØØØØ68Ø
      COMMON /GEINT/ XX, FN                                             ØØØØØ69Ø
      EXTERNAL FEINT                                                    ØØØØØ7ØØ
      XX = X                                                            ØØØØØ71Ø
      FN = N                                                            ØØØØØ72Ø
      SIG = 1.ØEØ                                                       ØØØØØ73Ø
      S = Ø.ØEØ                                                         ØØØØØ74Ø
      TOLA = TOL                                                        ØØØØØ75Ø
      B = X                                                             ØØØØØ76Ø
   1Ø CONTINUE                                                          ØØØØØ77Ø
      A = B                                                             ØØØØØ78Ø
      REL = TOL                                                         ØØØØØ79Ø
      B = B + SIG                                                       ØØØØØ8ØØ
      CALL GAUS8(FEINT, A, B, REL, ANS, IERR)                           ØØØØØ81Ø
      S = S + ANS                                                       ØØØØØ82Ø
      IF (ABS(ANS).LT.S*TOLA) GO TO 2Ø                                  ØØØØØ83Ø
      GO TO 1Ø                                                          ØØØØØ84Ø
   2Ø EINT = S*EXP((FN-1.ØEØ)*ALOG(X)-FLOAT(2-KODE)*X)                  ØØØØØ85Ø
      RETURN                                                            ØØØØØ86Ø
      END                                                               ØØØØØ87Ø


      FUNCTION FEINT(T)                                                 ØØØØØ88Ø
      COMMON /GEINT/ XX, FN                                             ØØØØØ89Ø
      FEINT = EXP(-T+XX-FN*ALOG(T))                                     ØØØØØ9ØØ
      RETURN                                                            ØØØØØ91Ø
      END                                                               ØØØØØ92Ø


      SUBROUTINE GAUS8  (FUN,A,B,ERR,ANS,IERR)                          ØØØØØ93Ø
C                                                                       ØØØØØ94Ø
C     BY RONDALL E JONES, SANDIA LABORATORIES                           ØØØØØ95Ø
C     SALIENT FEATURES -- INTERVAL BISECTION, COMBINED RELATIVE/ABSOLUTEØØØØØ96Ø
```

```
C     ERROR CONTROL, COMPUTED MAXIMUM REFINEMENT LEVEL WHEN A IS        ØØØØØ97Ø
C     CLOSE TO B.                                                       ØØØØØ98Ø
C                                                                       ØØØØØ99Ø
C     ABSTRACT                                                          ØØØØ1ØØØ
C        GAUS8 INTEGRATES REAL FUNCTIONS OF ONE VARIABLE OVER FINITE    ØØØØ1Ø1Ø
C        INTERVALS, USING AN ADAPTIVE 8-POINT LEGENDRE-GAUSS ALGORITHM. ØØØØ1Ø2Ø
C        GAUS8 IS INTENDED PRIMARILY FOR HIGH ACCURACY INTEGRATION      ØØØØ1Ø3Ø
C        OR INTEGRATION OF SMOOTH FUNCTIONS.  FOR LOWER ACCURACY        ØØØØ1Ø4Ø
C        INTEGRATION OF FUNCTIONS WHICH ARE NOT VERY SMOOTH,            ØØØØ1Ø5Ø
C        EITHER QNC3 OR QNC7 MAY BE MORE EFFICIENT.                     ØØØØ1Ø6Ø
C                                                                       ØØØØ1Ø7Ø
C     DESCRIPTION OF ARGUMENTS                                          ØØØØ1Ø8Ø
C                                                                       ØØØØ1Ø9Ø
C        INPUT--                                                        ØØØØ11ØØ
C        FUN - NAME OF EXTERNAL FUNCTION TO BE INTEGRATED.  THIS NAME   ØØØØ111Ø
C              MUST BE IN AN EXTERNAL STATEMENT IN THE CALLING PROGRAM. ØØØØ112Ø
C              FUN MUST BE A FUNCTION OF ONE REAL ARGUMENT.  THE VALUE  ØØØØ113Ø
C              OF THE ARGUMENT TO FUN IS THE VARIABLE OF INTEGRATION    ØØØØ114Ø
C              WHICH RANGES FROM A TO B.                                ØØØØ115Ø
C        A   - LOWER LIMIT OF INTEGRAL                                  ØØØØ116Ø
C        B   - UPPER LIMIT OF INTEGRAL (MAY BE LESS THAN A)             ØØØØ117Ø
C        ERR - IS A REQUESTED ERROR TOLERANCE.  NORMALLY PICK A VALUE OFØØØØ118Ø
C              ABS(ERR).LT.1.E-3.  ANS WILL NORMALLY HAVE NO MORE ERROR ØØØØ119Ø
C              THAN ABS(ERR) TIMES THE INTEGRAL OF THE ABSOLUTE VALUE   ØØØØ12ØØ
C              OF FUN(X).  USUALLY, SMALLER VALUES FOR ERR YIELD        ØØØØ121Ø
C              MORE ACCURACY AND REQUIRE MORE FUNCTION EVALUATIONS.     ØØØØ122Ø
C              A NEGATIVE VALUE FOR ERR CAUSES AN ESTIMATE OF THE       ØØØØ123Ø
C              ABSOLUTE ERROR IN ANS TO BE RETURNED IN ERR.             ØØØØ124Ø
C                                                                       ØØØØ125Ø
C        OUTPUT--                                                       ØØØØ126Ø
C        ERR - WILL BE AN ESTIMATE OF THE ERROR IN ANS IF THE INPUT     ØØØØ127Ø
C              VALUE OF ERR WAS NEGATIVE.  THE ESTIMATED ERROR IS SOLELYØØØØ128Ø
C              FOR INFORMATION TO THE USER AND SHOULD NOT BE USED AS    ØØØØ129Ø
C              A CORRECTION TO THE COMPUTED INTEGRAL.                   ØØØØ13ØØ
C        ANS - COMPUTED VALUE OF INTEGRAL                               ØØØØ131Ø
C        IERR- A STATUS CODE                                            ØØØØ132Ø
C            --NORMAL CODES                                             ØØØØ133Ø
C               1 ANS MOST LIKELY MEETS REQUESTED ERROR TOLERANCE,      ØØØØ134Ø
C                 OR A=B.                                               ØØØØ135Ø
C              -1 A AND B ARE TOO NEARLY EQUAL TO ALLOW NORMAL          ØØØØ136Ø
C                 INTEGRATION.  ANS IS SET TO ZERO.                     ØØØØ137Ø
C            --ABNORMAL CODE                                            ØØØØ138Ø
C               2 ANS PROBABLY DOES NOT MEET REQUESTED ERROR TOLERANCE. ØØØØ139Ø
C                                                                       ØØØØ14ØØ
C                                                                       ØØØØ141Ø
C                                                                       ØØØØ142Ø
C     GAUS8  USES SUBROUTINES ERRCHK, ERRGET, ERRPRT, ERXSET, ERSTGT    ØØØØ143Ø
C     COMPILE DECKS GAUS8, ERRCHK                                       ØØØØ144Ø
C                                                                       ØØØØ145Ø
      DIMENSION AA(3Ø),HH(3Ø),LR(3Ø),VL(3Ø),GR(3Ø)                      ØØØØ146Ø
      DATA X1,X2,X3,X4/Ø.18343 46424 9565Ø , Ø.52553 24Ø99 16329 ,      ØØØØ147Ø
     1                 Ø.79666 64774 13627 , Ø.96Ø28 98564 97536 /      ØØØØ148Ø
      DATA W1,W2,W3,W4/Ø.36268 37833 78362 , Ø.3137Ø 66458 77887 ,      ØØØØ149Ø
     1                 Ø.22238 1Ø344 53374 , Ø.1Ø122 85362 9Ø376 /      ØØØØ15ØØ
      DATA SQ2/1.41421356/,ICALL/Ø/                                     ØØØØ151Ø
      DATA NLMN/1/,NLMX/3Ø/,KMX/5ØØØ/,KML/6/,NBITS/48/                  ØØØØ152Ø
      G8(X,H) = H*( (W1*(FUN(X-X1*H)+FUN(X+X1*H))                       ØØØØ153Ø
     1              +W2*(FUN(X-X2*H)+FUN(X+X2*H)))                     ØØØØ154Ø
     2             +(W3*(FUN(X-X3*H)+FUN(X+X3*H))                      ØØØØ155Ø
     3              +W4*(FUN(X-X4*H)+FUN(X+X4*H))) )                   ØØØØ156Ø
C                                                                       ØØØØ157Ø
C     INITIALIZE                                                        ØØØØ158Ø
C                                                                       ØØØØ159Ø
      IF(ICALL.NE.Ø)CALL ERRCHK(-71,71H*****GAUS8 CALLED RECURSIVELY.  RØØØØ16ØØ
     1ECURSIVE CALLS ARE ILLEGAL IN FORTRAN. )                          ØØØØ161Ø
      ICALL = 1                                                         ØØØØ162Ø
      ANS = Ø.Ø                                                         ØØØØ163Ø
      IERR = 1                                                          ØØØØ164Ø
      CE = Ø.Ø                                                          ØØØØ165Ø
      IF (A.EQ.B) GO TO 35                                              ØØØØ166Ø
      LMX = NLMX                                                        ØØØØ167Ø
      LMN = NLMN                                                        ØØØØ168Ø
      IF (B.EQ.Ø.Ø) GO TO 4                                             ØØØØ169Ø
      IF (SIGN(1.Ø,B)*A.LE.Ø.Ø) GO TO 4                                 ØØØØ17ØØ
      C = ABS(1.Ø-A/B)                                                  ØØØØ171Ø
      IF (C.GT.Ø.1) GO TO 4                                             ØØØØ172Ø
```

```
      IF (C.LE.Ø.Ø) GO TO 35                                          ØØØØ173Ø
      NIB = Ø.5-ALOG(C)/ALOG(2.Ø)                                     ØØØØ174Ø
      LMX = MINØ(NLMX , NBITS-NIB-7)                                  ØØØØ175Ø
      IF (LMX.LT.1) GO TO 32                                          ØØØØ176Ø
      LMN = MINØ(LMN,LMX)                                             ØØØØ177Ø
    4 TOL = AMAX1(ABS(ERR),2.Ø**(5-NBITS))/2.Ø                        ØØØØ178Ø
      IF (ERR.EQ.Ø.Ø) TOL = Ø.5E-6                                    ØØØØ179Ø
      EPS = TOL                                                       ØØØØ18ØØ
      HH(1) = (B-A)/4.Ø                                               ØØØØ181Ø
      AA(1) = A                                                       ØØØØ182Ø
      LR(1) = 1                                                       ØØØØ183Ø
      L = 1                                                           ØØØØ184Ø
      EST = G8(AA(L)+2.Ø*HH(L),2.Ø*HH(L))                             ØØØØ185Ø
      K = 8                                                           ØØØØ186Ø
      AREA = ABS(EST)                                                 ØØØØ187Ø
      EF = Ø.5                                                        ØØØØ188Ø
      MXL = Ø                                                         ØØØØ189Ø
C                                                                     ØØØØ19ØØ
C     COMPUTE REFINED ESTIMATES, ESTIMATE THE ERROR, ETC.             ØØØØ191Ø
C                                                                     ØØØØ192Ø
    5 GL = G8(AA(L)+HH(L),HH(L))                                      ØØØØ193Ø
      GR(L) = G8(AA(L)+3.Ø*HH(L),HH(L))                               ØØØØ194Ø
      K = K+16                                                        ØØØØ195Ø
      AREA = AREA+(ABS(GL)+ABS(GR(L))-ABS(EST))                       ØØØØ196Ø
C     IF (L.LT.LMN) GO TO 11                                          ØØØØ197Ø
      GLR = GL+GR(L)                                                  ØØØØ198Ø
      EF = ABS(EST-GLR)*EF                                            ØØØØ199Ø
      AE = AMAX1(EPS*AREA,TOL*ABS(GLR))                               ØØØØ2ØØØ
      IF (EE-AE) 8,8,1Ø                                               ØØØØ2Ø1Ø
    7 MXL = 1                                                         ØØØØ2Ø2Ø
    8 CE = CE + (EST-GLR)                                             ØØØØ2Ø3Ø
      IF (LR(L)) 15,15,2Ø                                             ØØØØ2Ø4Ø
C                                                                     ØØØØ2Ø5Ø
C     CONSIDER THE LEFT HALF OF THIS LEVEL                            ØØØØ2Ø6Ø
C                                                                     ØØØØ2Ø7Ø
   1Ø IF (K.GT.KMX) LMX = KML                                         ØØØØ2Ø8Ø
      IF (L.GE.LMX) GO TO 7                                           ØØØØ2Ø9Ø
   11 L = L+1                                                         ØØØØ21ØØ
      EPS = EPS*Ø.5                                                   ØØØØ211Ø
      EF = EF/SQ2                                                     ØØØØ212Ø
      HH(L) = HH(L-1)*Ø.5                                             ØØØØ213Ø
      LR(L) = -1                                                      ØØØØ214Ø
      AA(L) = AA(L-1)                                                 ØØØØ215Ø
      EST = GL                                                        ØØØØ216Ø
      GO TO 5                                                         ØØØØ217Ø
C                                                                     ØØØØ218Ø
C     PROCEED TO RIGHT HALF AT THIS LEVEL                             ØØØØ219Ø
C                                                                     ØØØØ22ØØ
   15 VL(L) = GLR                                                     ØØØØ221Ø
   16 EST = GR(L-1)                                                   ØØØØ222Ø
      LR(L) = 1                                                       ØØØØ223Ø
      AA(L) = AA(L)+4.Ø*HH(L)                                         ØØØØ224Ø
      GO TO 5                                                         ØØØØ225Ø
C                                                                     ØØØØ226Ø
C     RETURN ONE LEVEL                                                ØØØØ227Ø
C                                                                     ØØØØ228Ø
   2Ø VR = GLR                                                        ØØØØ229Ø
   22 IF (L.LE.1) GO TO 3Ø                                            ØØØØ23ØØ
      L = L-1                                                         ØØØØ231Ø
      EPS = EPS*2.Ø                                                   ØØØØ232Ø
      EF = EF*SQ2                                                     ØØØØ233Ø
      IF (LR(L)) 24,24,26                                             ØØØØ234Ø
   24 VL(L) = VL(L+1)+VR                                              ØØØØ235Ø
      GO TO 16                                                        ØØØØ236Ø
   26 VR = VL(L+1)+VR                                                 ØØØØ237Ø
      GO TO 22                                                        ØØØØ238Ø
C                                                                     ØØØØ239Ø
C      EXIT                                                           ØØØØ24ØØ
C                                                                     ØØØØ241Ø
   3Ø ANS = VR                                                        ØØØØ242Ø
      IF ((MXL.EQ.Ø).OR.(ABS(CE).LE.2.Ø*TOL*AREA)) GO TO 35           ØØØØ243Ø
      IERR = 2                                                        ØØØØ244Ø
      CALL ERRCHK(51,51HIN GAUS8 , ANS IS PROBABLY INSUFFICIENTLY ACCURAØØØØ245Ø
     1TE.)                                                            ØØØØ246Ø
      GO TO 35                                                        ØØØØ247Ø
   32 IERR =-1                                                        ØØØØ248Ø
```

```
      CALL ONECHK(-7Ø,7ØHTHE FOLLOWING TEMPORARY INFORMATIVE DIAGNOSTIC Ø0002490
     + WILL APPEAR ONLY ONCE. )                                        Ø0002500
      CALLONECHK(-1Ø2,1Ø2HIN GAUS8 , A AND B ARE TOO NEARLY EQUAL TO ALLØ0002510
     1OW NORMAL INTEGRATION.  ANS IS SET TO ZERO, AND IERR=-1.)        Ø0002520
   35 ICALL = Ø                                                        Ø0002530
      IF (ERR.LT.Ø.Ø) ERR = CE                                         Ø0002540
      RETURN                                                           Ø0002550
      END                                                              Ø0002560


      SUBROUTINE ERRCHK(NCHARS,NARRAY)                                 Ø0002570
C                                                                      Ø0002580
C     SANDIA MATHEMATICAL PROGRAM LIBRARY                              Ø0002590
C     APPLIED MATHEMATICS DIVISION 2642                                Ø0002600
C     SANDIA LABORATORIES                                              Ø0002610
C     ALBUQUERQUE, NEW MEXICO 87115                                    Ø0002620
C                                                                      Ø0002630
C     SIMPLIFIED VERSION FOR STAND-ALONE USE.     APRIL 1977           Ø0002640
C                                                                      Ø0002650
C     ABSTRACT                                                         Ø0002660
C         THE ROUTINES ERRCHK, ERXSET, AND ERRGET TOGETHER PROVIDE     Ø0002670
C         A UNIFORM METHOD WITH SEVERAL OPTIONS FOR THE PROCESSING     Ø0002680
C         OF DIAGNOSTICS AND WARNING MESSAGES WHICH ORIGINATE          Ø0002690
C         IN THE MATHEMATICAL PROGRAM LIBRARY ROUTINES.                Ø0002700
C         ERRCHK IS THE CENTRAL ROUTINE, WHICH ACTUALLY PROCESSES      Ø0002710
C         MESSAGES.                                                    Ø0002720
C                                                                      Ø0002730
C     DESCRIPTION OF ARGUMENTS                                         Ø0002740
C         NCHARS - NUMBER OF CHARACTERS IN HOLLERITH MESSAGE.          Ø0002750
C                  IF NCHARS IS NEGATED, ERRCHK WILL UNCONDITIONALLY   Ø0002760
C                  PRINT THE MESSAGE AND STOP EXECUTION.  OTHERWISE,   Ø0002770
C                  THE BEHAVIOR OF ERRCHK MAY BE CONTROLLED BY         Ø0002780
C                  AN APPROPRIATE CALL TO ERXSET.                      Ø0002790
C         NARRAY - NAME OF ARRAY OR VARIABLE CONTAINING THE MESSAGE,   Ø0002800
C                  OR ELSE A LITERAL HOLLERITH CONSTANT CONTAINING     Ø0002810
C                  THE MESSAGE.  BY CONVENTION, ALL MESSAGES SHOULD    Ø0002820
C                  BEGIN WITH *IN SUBNAM, ...*, WHERE SUBNAM IS THE    Ø0002830
C                  NAME OF THE ROUTINE CALLING ERRCHK.                 Ø0002840
C                                                                      Ø0002850
C     EXAMPLES                                                         Ø0002860
C         1. TO ALLOW CONTROL BY CALLING ERXSET, USE                   Ø0002870
C            CALL ERRCHK(3Ø,3ØHIN QUAD, INVALID VALUE OF ERR.)         Ø0002880
C         2. TO UNCONDITIONALLY PRINT A MESSAGE AND STOP EXECUTION, USE Ø0002890
C            CALL ERRCHK(-3Ø,3ØHIN QUAD, INVALID VALUE OF ERR.)        Ø0002900
C                                                                      Ø0002910
C                                                                      Ø0002920
C                                                                      Ø0002930
C     ERRCHK USES SUBROUTINES ERRGET, ERRPRT, ERXSET, ERSTGT           Ø0002940
C     COMPILE DECKS ERRCHK                                             Ø0002950
C                                                                      Ø0002960
      DIMENSION NARRAY(14)                                             Ø0002970
C                                                                      Ø0002980
      IOUT=6                                                           Ø0002990
      CALL ERRGET(NF,NT)                                               Ø0003000
C     IF ERRCHK WAS CALLED WITH NEGATIVE CHARACTER COUNT, SET FATAL FLAGØ0003010
      IF (NCHARS.LT.Ø) NF = -1                                         Ø0003020
C     IF MESSAGES ARE TO BE SUPPRESSED, RETURN                         Ø0003030
      IF (NF.EQ.Ø) RETURN                                              Ø0003040
C     IF CHARACTER COUNT IS INVALID, STOP                              Ø0003050
C     IF (NCHARS.EQ.Ø) PRINT 5                                         Ø0003060
      IF (NCHARS .EQ. Ø) WRITE (IOUT,5)                                Ø0003070
    5 FORMAT(/31H ERRCHK WAS CALLED INCORRECTLY.)                      Ø0003080
      IF (NCHARS.EQ.Ø) STOP                                            Ø0003090
C     PRINT MESSAGE                                                    Ø0003100
      CALL ERRPRT(IABS(NCHARS),NARRAY)                                 Ø0003110
C     IF LAST MESSAGE, SAY SO                                          Ø0003120
C     IF (NF.EQ.1) PRINT 1Ø                                            Ø0003130
      IF (NF .EQ. 1) WRITE (IOUT,1Ø)                                   Ø0003140
   1Ø FORMAT (3ØH ERRCHK MESSAGE LIMIT REACHED.)                       Ø0003150
C     PRINT TRACE-BACK IF ASKED TO                                     Ø0003160
C     IF ((NT.GT.Ø).OR.(NF.LT.Ø)) CALL SYSTEM ROUTINE FOR TRACEBACK    Ø0003170
C     DECREMENT MESSAGE COUNT                                          Ø0003180
      IF (NF.GT.Ø) NF = NF-1                                           Ø0003190
      CALL ERXSET(NF,NT)                                               Ø0003200
C     IF ALL IS WELL, RETURN                                           Ø0003210
      IF (NF.GE.Ø) RETURN                                              Ø0003220
```

```
C     IF THIS MESSAGE IS SUPPRESSABLE BY AN ERXSET CALL,                ØØØØ323Ø
C     THEN EXPLAIN ERXSET USAGE.                                        ØØØØ324Ø
C     IF (NCHARS.GT.Ø) PRINT 15                                         ØØØØ325Ø
      IF (NCHARS .GT. Ø) WRITE (IOUT,15)                                ØØØØ326Ø
   15 FORMAT (/13H *** NOTE ***                                         ØØØØ327Ø
     1/53H TO MAKE THE ERROR MESSAGE PRINTED ABOVE BE NONFATAL,         ØØØØ328Ø
     2/39H OR TO SUPPRESS THE MESSAGE COMPLETELY,                       ØØØØ329Ø
     3/37H INSERT AN APPROPRIATE CALL TO ERXSET                         ØØØØ33ØØ
     4,3ØH AT THE START OF YOUR PROGRAM.                                ØØØØ331Ø
     5/62H FOR EXAMPLE, TO PRINT UP TO 1Ø NONFATAL WARNING MESSAGES, USEØØØØ332Ø
     6/27H          CALL ERXSET(1Ø,Ø)     )                            ØØØØ333Ø
C     PRINT 2Ø                                                          ØØØØ334Ø
      WRITE (IOUT,2Ø)                                                   ØØØØ335Ø
   2Ø FORMAT (/28H PROGRAM ABORT DUE TO ERROR.)                         ØØØØ336Ø
      STOP                                                              ØØØØ337Ø
      END                                                               ØØØØ338Ø


      SUBROUTINE ONECHK(NCHARS,NARRAY)                                  ØØØØ339Ø
C                                                                       ØØØØ34ØØ
C     ABSTRACT                                                          ØØØØ341Ø
C         ONECHK IS A COMPANION ROUTINE OF ERRCHK.  IT IS CALLED        ØØØØ342Ø
C         JUST LIKE ERRCHK, AND MESSAGES FROM IT MAY BE SUPPRESSED      ØØØØ343Ø
C         BY AN APPROPRIATE CALL TO ERXSET.  IT DIFFERS FROM ERRCHK     ØØØØ344Ø
C         IN THAT EACH CALL TO ONECHK WILL PRODUCE NO MORE THAN ONE     ØØØØ345Ø
C         PRINTED MESSAGE, REGARDLESS OF HOW MANY TIMES THAT CALL IS    ØØØØ346Ø
C         EXECUTED, AND ONECHK NEVER TERMINATES EXECUTION.              ØØØØ347Ø
C         ITS PURPOSE IS TO PROVIDE ONE-TIME-ONLY INFORMATIVE           ØØØØ348Ø
C         DIAGNOSTICS.                                                  ØØØØ349Ø
C                                                                       ØØØØ35ØØ
C     DESCRIPTION OF ARGUMENTS                                          ØØØØ351Ø
C         NCHARS - NUMBER OF CHARACTERS IN THE MESSAGE.                 ØØØØ352Ø
C                  IF NEGATED, THE MESSAGE WILL BE PRINTED (ONCE) EVEN  ØØØØ353Ø
C                  IF NFATAL HAS BEEN SET TO Ø (SEE ERXSET).            ØØØØ354Ø
C         NARRAY - SAME AS IN ERRCHK                                    ØØØØ355Ø
C                                                                       ØØØØ356Ø
C                                                                       ØØØØ357Ø
C                                                                       ØØØØ358Ø
C     ONECHK USES SUBROUTINES ERRGET, ERRPRT, ERXSET, ERSTGT            ØØØØ359Ø
C     COMPILE DECKS ERRCHK                                              ØØØØ36ØØ
C                                                                       ØØØØ361Ø
      DIMENSION NARRAY(14)                                              ØØØØ362Ø
      DATA NFLAG/4H.$,*/                                                ØØØØ363Ø
      IF (NARRAY(1).EQ.NFLAG) RETURN                                    ØØØØ364Ø
      CALL ERRGET(NF,NT)                                                ØØØØ365Ø
      IF ((NF.EQ.Ø).AND.(NCHARS.GT.Ø)) RETURN                           ØØØØ366Ø
      CALL ERRPRT (59,59HTHE FOLLOWING INFORMATIVE DIAGNOSTIC WILL APPEAØØØØ367Ø
     1R ONLY ONCE.)                                                     ØØØØ368Ø
      CALL ERRPRT(IABS(NCHARS),NARRAY)                                  ØØØØ369Ø
      IF (NF.GT.Ø) NF = NF-1                                            ØØØØ37ØØ
      CALL ERXSET(NF,NT)                                                ØØØØ371Ø
      NARRAY(1) = NFLAG                                                 ØØØØ372Ø
      RETURN                                                            ØØØØ373Ø
      END                                                               ØØØØ374Ø


      SUBROUTINE ERRPRT(NCHARS,NARRAY)                                  ØØØØ375Ø
C                                                                       ØØØØ376Ø
C     UTILITY ROUTINE TO SIMPLY PRINT THE HOLLERITH MESSAGE IN NARRAY,  ØØØØ377Ø
C     WHOSE LENGTH IS NCHARS CHARACTERS.                                ØØØØ378Ø
C                                                                       ØØØØ379Ø
      DIMENSION NARRAY(14)                                              ØØØØ38ØØ
C                                                                       ØØØØ381Ø
C     NOTE - NCH MUST BE THE NUMBER OF HOLLERITH CHARACTERS STORED      ØØØØ382Ø
C     PER WORD.  IF NCH IS CHANGED, FORMAT 1 MUST ALSO BE               ØØØØ383Ø
C     CHANGED CORRESPONDINGLY.                                          ØØØØ384Ø
C                                                                       ØØØØ385Ø
      IOUT=6                                                            ØØØØ386Ø
      NCH = 1Ø                                                          ØØØØ387Ø
C     FOR LINE PRINTERS, USE                                            ØØØØ388Ø
    1 FORMAT (1X,13A1Ø)                                                 ØØØØ389Ø
C     FOR DATA TERMINALS, USE                                           ØØØØ39ØØ
C   1 FORMAT (1X,7A1Ø)                                                  ØØØØ391Ø
      NWORDS = (NCHARS+NCH-1)/NCH                                       ØØØØ392Ø
C     PRINT 1,(NARRAY(I),I=1,NWORDS)                                    ØØØØ393Ø
```

```
      WRITE (IOUT,1) (NARRAY(I),I=1,NWORDS)                              ØØØØ394Ø
      RETURN                                                             ØØØØ395Ø
      END                                                                ØØØØ396Ø


      SUBROUTINE ERXSET(NFATAL,NTRACE)                                   ØØØØ397Ø
C                                                                        ØØØØ398Ø
C     ABSTRACT                                                           ØØØØ399Ø
C         ERXSET IS A COMPANION ROUTINE TO SUBROUTINE ERRCHK.            ØØØØ4ØØØ
C         ERXSET ASSIGNS THE VALUES OF NFATAL AND NTRACE RESPECTIVELY    ØØØØ4Ø1Ø
C         TO NF AND NT IN COMMON BLOCK MLBLKØ THEREBY SPECIFYING THE     ØØØØ4Ø2Ø
C         STATE OF THE OPTIONS WHICH CONTROL THE EXECUTION OF ERRCHK.    ØØØØ4Ø3Ø
C                                                                        ØØØØ4Ø4Ø
C     DESCRIPTION OF ARGUMENTS                                           ØØØØ4Ø5Ø
C         BOTH ARGUMENTS ARE INPUT ARGUMENTS OF DATA TYPE INTEGER.       ØØØØ4Ø6Ø
C         NFATAL - IS A FATAL-ERROR / MESSAGE-LIMIT FLAG. A NEGATIVE     ØØØØ4Ø7Ø
C                  VALUE DENOTES THAT DETECTED DIFFICULTIES ARE TO BE    ØØØØ4Ø8Ø
C                  TREATED AS FATAL ERRORS.  NONNEGATIVE MEANS NONFATAL. ØØØØ4Ø9Ø
C                  A NONNEGATIVE VALUE IS THE MAXIMUM NUMBER OF NONFATAL ØØØØ41ØØ
C                  WARNING MESSAGES WHICH WILL BE PRINTED BY ERRCHK,     ØØØØ411Ø
C                  AFTER WHICH NONFATAL MESSAGES WILL NOT BE PRINTED.    ØØØØ412Ø
C                  (DEFAULT VALUE IS -1.)                                ØØØØ413Ø
C         NTRACE - .GE.1 WILL CAUSE A TRACE-BACK TO BE GIVEN,            ØØØØ414Ø
C                        IF THIS FEATURE IS IMPLEMENTED ON THIS SYSTEM.  ØØØØ415Ø
C                  .LE.Ø WILL SUPPRESS ANY TRACE-BACK, EXCEPT FOR        ØØØØ416Ø
C                        CASES WHEN EXECUTION IS TERMINATED.             ØØØØ417Ø
C                  (DEFAULT VALUE IS Ø.)                                 ØØØØ418Ø
C                                                                        ØØØØ419Ø
C         *NOTE* -- SOME CALLS TO ERRCHK WILL CAUSE UNCONDITIONAL        ØØØØ42ØØ
C         TERMINATION OF EXECUTION.  ERXSET HAS NO EFFECT ON SUCH CALLS. ØØØØ421Ø
C                                                                        ØØØØ422Ø
C     EXAMPLES                                                           ØØØØ423Ø
C         1. TO PRINT UP TO 1ØØ MESSAGES AS NONFATAL WARNINGS USE        ØØØØ424Ø
C            CALL ERXSET(1ØØ,Ø)                                          ØØØØ425Ø
C         2. TO SUPPRESS ALL MATHLIB WARNING MESSAGES USE                ØØØØ426Ø
C            CALL ERXSET(Ø,Ø)                                            ØØØØ427Ø
C                                                                        ØØØØ428Ø
C                                                                        ØØØØ429Ø
C                                                                        ØØØØ43ØØ
C     ERXSET USES SUBROUTINES ERSTGT                                     ØØØØ431Ø
C     COMPILE DECKS ERRCHK                                               ØØØØ432Ø
C                                                                        ØØØØ433Ø
      CALL ERSTGT(Ø,NFATAL,NTRACE)                                       ØØØØ434Ø
      RETURN                                                             ØØØØ435Ø
      END                                                                ØØØØ436Ø


      SUBROUTINE ERRGET(NFATAL,NTRACE)                                   ØØØØ437Ø
C                                                                        ØØØØ438Ø
C     ABSTRACT                                                           ØØØØ439Ø
C         ERRGET IS A COMPANION ROUTINE TO SUBROUTINE ERRCHK.            ØØØØ44ØØ
C         ERRGET ASSIGNS TO NFATAL AND NTRACE RESPECTIVELY THE VALUES    ØØØØ441Ø
C         OF NF AND NT IN COMMON BLOCK MLBLKØ THEREBY ASCERTAINING THE   ØØØØ442Ø
C         STATE OF THE OPTIONS WHICH CONTROL THE EXECUTION OF ERRCHK.    ØØØØ443Ø
C                                                                        ØØØØ444Ø
C     DESCRIPTION OF ARGUMENTS                                           ØØØØ445Ø
C     DESCRIPTION OF ARGUMENTS                                           ØØØØ445Ø
C         BOTH ARGUMENTS ARE OUTPUT ARGUMENTS OF DATA TYPE INTEGER.      ØØØØ446Ø
C         NFATAL - CURRENT VALUE OF NF (SEE DESCRIPTION OF ERXSET.)      ØØØØ447Ø
C         NTRACE - CURRENT VALUE OF NT (SEE DESCRIPTION OF ERXSET.)      ØØØØ448Ø
C                                                                        ØØØØ449Ø
      CALL ERSTGT(1,NFATAL,NTRACE)                                       ØØØØ45ØØ
      RETURN                                                             ØØØØ451Ø
      END                                                                ØØØØ452Ø


      SUBROUTINE ERSTGT(K,NFATAL,NTRACE)                                 ØØØØ453Ø
C                                                                        ØØØØ454Ø
C     THIS ROUTINE IS A SLAVE TO ERRGET AND ERRSET WHICH KEEPS           ØØØØ455Ø
C     THE FLAGS AS LOCAL VARIABLES.                                      ØØØØ456Ø
C                                                                        ØØØØ457Ø
C     *** IF LOCAL VARIABLES ARE NOT NORMALLY RETAINED BETWEEN           ØØØØ458Ø
C     CALLS ON THIS SYSTEM, THE VARIABLES LNF AND LNT CAN BE             ØØØØ459Ø
C     PLACED IN A COMMON BLOCK AND PRESET TO THE FOLLOWING               ØØØØ46ØØ
```

```
C     VALUES IN THE MAIN PROGRAM.                                  ØØØØ461Ø
C                                                                  ØØØØ462Ø
      DATA LNF/-1/,LNT/Ø/                                          ØØØØ463Ø
      IF (K.LE.Ø) LNF = NFATAL                                     ØØØØ464Ø
      IF (K.LE.Ø) LNT = NTRACE                                     ØØØØ465Ø
      IF (K.GT.Ø) NFATAL = LNF                                     ØØØØ466Ø
      IF (K.GT.Ø) NTRACE = LNT                                     ØØØØ467Ø
      RETURN                                                       ØØØØ468Ø
      END                                                          ØØØØ469Ø
```

ALGORITHM 557
PAGP, A Partitioning Algorithm for (Linear) Goal Programming Problems [H]

JEFFREY L. ARTHUR
Oregon State University
and
A. RAVINDRAN
Purdue University

Key Words and Phrases: goal program, multiple objective optimization, constraint partitioning, simplex method
CR Categories: 5.41
Language: Fortran

DESCRIPTION

This algorithm is a Fortran implementation of the procedures developed in [1].

REFERENCE

1. ARTHUR, J.L., AND RAVINDRAN, A. PAGP, A partitioning algorithm for (linear) goal programming problems. *ACM Trans. Math. Softw. 6,* 3 (Sept. 1980), 378-386.

ALGORITHM

```
C      PROGRAM MAIN (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)                A   1Ø
C ***********************************************************************
C ****
C ****
C **** PAGP (THE PARTITIONING ALGORITHM FOR GOAL PROGRAMMING) IS
C **** DESIGNED TO SOLVE THE LINEAR GOAL PROGRAMMING PROBLEM.  THE
C **** ALGORITHM PARTITIONS THE GOAL CONSTRAINTS OF THE PROBLEM
C **** ACCORDING TO THE HIGHEST PRIORITY ASSIGNED TO EITHER DEVIATIONAL
C **** VARIABLE (D- OR D+) IN EACH GOAL.  THE ALGORITHM IS THUS ABLE TO
C **** SOLVE A SEQUENCE OF SMALLER PROBLEMS IN ORDER TO FIND A SOLUTION
C **** TO THE ORIGINAL PROBLEM.
C ****
C ****
C ***********************************************************************
C ****
C ****
C **** MUCH OF THE NOTATION AND STRUCTURE OF THE PAGP CODE IS TAKEN
C **** (WITH THE AUTHOR'S PERMISSION) FROM THE LINEAR GOAL PROGRAMMING
```

Received 24 April 1978; revised 21 June 1979; accepted 12 December 1979.

Authors' addresses: J.L. Arthur, Department of Statistics, Oregon State University, Corvallis, OR 97331; A. Ravindran, School of Industrial Engineering, Purdue University, West Lafayette, IN 47907.

```
C **** CODE GIVEN IN
C ****
C ****           IGNIZIO,J.P., GOAL PROGRAMMING AND EXTENSIONS,
C ****                         ------------------------------
C ****           D.C.HEATH AND CO., LEXINGTON,MA(1976)
C ****
C ****
C ***************************************************************
C ****
C ****
C **** THE CODE FOR PAGP USES THE FOLLOWING ARRAYS-
C ****
C ****         TE(NROW,NCOL) = THE COEFFICIENT OF THE VARIABLE IN
C ****                         COLUMN NCOL IN THE CONSTRAINT IN ROW NROW
C ****
C ****         TT(NP,NCOL)   = THE WEIGHT OF THE VARIABLE IN COLUMN NCOL
C ****                         AT PRIORITY NP
C ****
C ****         TB(NROW)      = THE RIGHT HAND SIDE CONSTANT OF THE
C ****                         CONSTRAINT IN ROW NROW
C ****
C ****         TL(NROW,NP)   = THE WEIGHT ASSIGNED TO THE BASIC VARIABLE
C ****                         IN ROW NROW AT PRIORITY NP
C ****
C ****         TI(NP,NCOL)   = THE RELATIVE WEIGHT OF THE VARIABLE IN
C ****                         COLUMN NCOL AT PRIORITY NP
C ****
C ****         TA(NP)        = THE TOTAL DEVIATION FROM THE GOALS AT
C ****                         PRIORITY NP
C ****
C ****         NC(NP)        = THE NUMBER OF GOAL CONSTRAINTS ASSIGNED
C ****                         TO PRIORITY NP BY THE PARTITION
C ****
C ****         NCON(I,NP)    = THE SUBSCRIPT OF THE I-TH CONSTRAINT
C ****                         ASSIGNED TO PRIORITY NP (I=1,...,NC(NP))
C ****
C ****         NTOF(NP)      = THE NUMBER OF TERMS IN THE OBJECTIVE
C ****                         FUNCTION AT PRIORITY NP
C ****
C ****         IND(NCOL)     = 1, IF THE VARIABLE IN COLUMN NCOL IS
C ****                         ELIGIBLE TO ENTER THE BASIS
C ****                       = Ø, OTHERWISE
C ****
C ****         JROW(NROW,1)  = THE TYPE OF BASIC VARIABLE IN ROW NROW,
C ****                         WHERE TYPE IS GIVEN BELOW
C ****
C ****                                 TYPE      JROW( . ,1)
C ****                                 ****      ***********
C ****                                  X             2
C ****                                  D+            3
C ****                                  D-            4
C ****
C ****         JROW(NROW,2)  = THE SUBSCRIPT OF THE BASIC VARIABLE IN
C ****                         ROW NROW
C ****
C ****         JCOL(NCOL,1)  = THE TYPE OF VARIABLE IN COLUMN NCOL
C ****                         (TYPE IS DEFINED AS ABOVE)
C ****
C ****         JCOL(NCOL,2)  = THE SUBSCRIPT OF THE VARIABLE IN
C ****                         COLUMN NCOL
C ****
C ****
C ***************************************************************
C ****
C ****
C **** AS PRESENTLY DIMENSIONED, THE PAGP CODE CAN SOLVE PROBLEMS WITH
C **** UP TO 6Ø CONSTRAINTS (BOTH REAL + GOAL CONSTRAINTS), 125
C **** VARIABLES (DECISION + DEVIATIONAL VARIABLES), AND 1Ø PRIORITIES.
C ****
C ****
C ***************************************************************
C ****
C ****
      COMMON TT(1Ø,125),TB(6Ø),TE(6Ø,125),TL(6Ø,1Ø),TA(1Ø),TI(1Ø,125),JC  A   2Ø
     1OL(125,2),NCOLI,NROWI,NPRIC,NC(1Ø),JROW(6Ø,2),NVAR,NPRIT,IND(125)   A   3Ø
      COMMON /PHASE1/ W,NRCON,NDVR                                        A   4Ø
```

```
      COMMON /CHNG/ NCON(6Ø,1Ø),NTOF(1Ø)                                A   5Ø
      INTEGER ALTST                                                     A   6Ø
C                                                                       A   8Ø
C **** READ IN PROBLEM DATA                                             A   9Ø
C ****
C **** NPRIT=THE TOTAL NUMBER OF PRIORITIES
C ****
C **** NVAR=THE TOTAL NUMBER OF DECISION VARIABLES(INCLUDING SLACK AND
C ****       SURPLUS VARIABLES BUT EXCLUDING ARTIFICIAL VARIABLES FOR THE
C ****       REAL CONSTRAINTS)
C ****
C **** NRCON=THE NUMBER OF REAL CONSTRAINTS
C                                                                       A  1ØØ
      READ (5,12Ø) NPRIT,NVAR,NRCON                                     A  11Ø
      READ (5,121) (NC(NP),NP=1,NPRIT)                                  A  12Ø
      DO 1Ø1 NP=1,NPRIT                                                 A  13Ø
         IF (NC(NP).EQ.Ø) GO TO 1Ø1                                     A  14Ø
         NCTMP=NC(NP)
         READ (5,122) (NCON(N,NP),N=1,NCTMP)
 1Ø1 CONTINUE                                                           A  16Ø
      READ (5,121) (NTOF(NP),NP=1,NPRIT)                                A  17Ø
C                                                                       A  18Ø
C **** INITIALIZE SUBPROBLEM DIMENSIONS AND COLUMN INDICATORS.          A  19Ø
C ****
C **** NCOLI=THE NUMBER OF COLUMNS IN THE CURRENT WORKING TABLEAU
C ****
C **** NROWI=THE NUMBER OF ROWS IN THE CURRENT WORKING TABLEAU
C ****
C **** NPRIC=THE PRIORITY CURRENTLY BEING OPTIMIZED
C ****
C **** ZERO THE TE, TL, TT, AND TI ARRAYS.                              A  2ØØ
C                                                                       A  21Ø
      NCOLI=Ø                                                           A  22Ø
      NROWI=Ø                                                           A  23Ø
      NPRIC=Ø                                                           A  24Ø
      DO 1Ø4 NCR=1,125                                                  A  25Ø
         IND(NCR)=1                                                     A  26Ø
         DO 1Ø2 NR=1,6Ø                                                 A  27Ø
 1Ø2     TE(NR,NCR)=Ø.                                                  A  28Ø
         DO 1Ø3 NP=1,1Ø                                                 A  29Ø
            TI(NP,NCR)=Ø.                                               A  3ØØ
 1Ø3     TT(NP,NCR)=Ø.                                                  A  31Ø
 1Ø4 CONTINUE                                                           A  32Ø
      DO 1Ø5 NR=1,6Ø                                                    A  33Ø
      DO 1Ø5 NP=1,1Ø                                                    A  34Ø
 1Ø5 TL(NR,NP)=Ø.                                                       A  35Ø
C                                                                       A  36Ø
C **** CHECK FOR REAL CONSTRAINTS.                                      A  37Ø
C                                                                       A  38Ø
      IF (NRCON.EQ.Ø) GO TO 1Ø6                                         A  39Ø
      CALL PHSE1                                                        A  4ØØ
      IF (NDVR.LE.Ø) GO TO 116                                          A  41Ø
      IF (W.GT.Ø.) GO TO 117                                            A  42Ø
C                                                                       A  43Ø
C **** THE PARTITIONING ALGORITHM BEGINS.                               A  44Ø
C                                                                       A  45Ø
 1Ø6 NPRIC=NPRIC+1                                                      A  46Ø
      IF (NPRIC.EQ.1.AND.NRCON.EQ.Ø) GO TO 1Ø7                          A  47Ø
      GO TO 1Ø8                                                         A  48Ø
 1Ø7 CALL READ1                                                         A  49Ø
      GO TO 1Ø9                                                         A  5ØØ
 1Ø8 CALL READ2                                                         A  51Ø
 1Ø9 CALL CINDX                                                         A  52Ø
      CALL TEST (NEVC,NDVR)                                             A  53Ø
C                                                                       A  54Ø
C **** IF NEVC IS LESS THAN ZERO, THE SUBPROBLEM IS OPTIMIZED.          A  55Ø
C                                                                       A  56Ø
      IF (NEVC.LE.Ø) GO TO 11Ø                                          A  57Ø
C                                                                       A  58Ø
C **** IF NDVR IS LESS THAN ZERO, NO MINIMUM POSITIVE RATIO WAS FOUND.  A  59Ø
C                                                                       A  6ØØ
      IF (NDVR.LE.Ø) GO TO 116                                          A  61Ø
      CALL PERM (NEVC,NDVR)                                             A  62Ø
      GO TO 1Ø9                                                         A  63Ø
C                                                                       A  64Ø
C **** IF THERE ARE NO MORE PRIORITIES, TOTAL PROBLEM IS OPTIMIZED.      A  65Ø
```

```
C **** PRINT THE OPTIMAL SOLUTION.                                        A  66Ø
C                                                                         A  67Ø
  11Ø IF (NPRIC.EQ.NPRIT) GO TO 115                                       A  68Ø
C                                                                         A  69Ø
C **** SINCE THERE ARE MORE PRIORITIES, MOVE ON TO THE NEXT SUBPROBLEM    A  7ØØ
C **** IF THERE ARE ALTERNATE SOLUTIONS.  FIRST, ELIMINATE THOSE          A  71Ø
C **** COLUMNS WHICH CAN NOT ENTER THE BASIS.  IF THERE ARE NO            A  72Ø
C **** ALTERNATE SOLUTIONS, PRINT THE UNIQUE OPTIMAL SOLUTION.            A  73Ø
C                                                                         A  74Ø
      ALTST=Ø                                                             A  75Ø
      DO 112 NCR=1,NCOLI                                                  A  76Ø
         IF (IND(NCR).EQ.Ø) GO TO 112                                     A  77Ø
         IF (TI(NPRIC,NCR).GT.Ø.) GO TO 112                               A  78Ø
         DO 111 NR=1,NROWI                                                A  79Ø
            IF (JROW(NR,1).EQ.JCOL(NCR,1).AND.JROW(NR,2).EQ.JCOL(NCR,2))  A  8ØØ
     1       GO TO 112                                                    A  81Ø
  111    CONTINUE                                                         A  82Ø
         ALTST=1                                                          A  83Ø
  112 CONTINUE                                                            A  84Ø
C                                                                         A  85Ø
C **** IF ALTST=1, THERE ARE ALTERNATE SOLUTIONS.                         A  86Ø
C                                                                         A  87Ø
      IF (ALTST.EQ.1) GO TO 113                                           A  88Ø
      GO TO 115                                                           A  89Ø
C                                                                         A  9ØØ
C **** ELIMINATE THOSE COLUMNS WITH A POSITIVE RELATIVE COST AT           A  91Ø
C **** PRIORITY NPRIC.                                                    A  92Ø
C                                                                         A  93Ø
  113 DO 114 NCR=1,NCOLI                                                  A  94Ø
  114 IF (TI(NPRIC,NCR).GT.Ø.) IND(NCR)=Ø                                 A  95Ø
      GO TO 1Ø6                                                           A  96Ø
C
C **** THE OPTIMIZATION IS OVER. PRINT OUT THE FINAL SOLUTION.
C
  115 CALL POUT                                                           A  97Ø
      GO TO 119                                                           A  98Ø
  116 WRITE (6,123) NPRIC                                                 A  99Ø
      GO TO 119                                                           A 1ØØØ
  117 WRITE (6,124) W                                                     A 1Ø1Ø
      WRITE (6,125)                                                       A 1Ø2Ø
      DO 118 NR=1,NROWI                                                   A 1Ø3Ø
         WRITE (6,126) JROW(NR,1),JROW(NR,2),TB(NR)                       A 1Ø4Ø
  118 CONTINUE                                                            A 1Ø5Ø
  119 STOP
C                                                                         A 112Ø
  12Ø FORMAT (3I5)                                                        A 113Ø
  121 FORMAT (1ØI5)                                                       A 114Ø
  122 FORMAT (16I5)                                                       A 115Ø
  123 FORMAT (/  4ØH   THE PROGRAM TERMINATED ON SUBPROBLEM ,I4, 42H  NO
     1 MINIMUM POSITIVE RATIO COULD BE FOUND)                             A 117Ø
  124 FORMAT (/  65H  THE PROGRAM TERMINATED IN PHASE 1 WITH OBJECTIVE F
     1UNCTION VALUE,F15.4)                                                A 119Ø
  125 FORMAT (/  55H  THE OPTIMAL SOLUTION TO THE PHASE 1 PROBLEM IS
     1    //   6H  TYPE,2X,  3HSUB,8X,  5HVALUE)
  126 FORMAT (2I5,F15.4)                                                  A 122Ø
C                                                                         A 126Ø
      END                                                                 A 127Ø


      SUBROUTINE PHSE1                                                    B   1Ø
C
C **** SUBROUTINE PHSE1 READS IN ANY REAL CONSTRAINTS AND PERFORMS A
C **** PHASE 1 SIMPLEX PROCEDURE IN ORDER TO FIND AN INITIAL BASIC
C **** FEASIBLE SOLUTION.
C
      COMMON TT(1Ø,125),TB(6Ø),TE(6Ø,125),TL(6Ø,1Ø),TA(1Ø),TI(1Ø,125),JC   B   2Ø
     1OL(125,2),NCOLI,NROWI,NPRIC,NC(1Ø),JROW(6Ø,2),NVAR,NPRIT,IND(125)    B   3Ø
      COMMON /PHASE1/ W,NRCON,NDVR                                        B   4Ø
      DIMENSION C(125), CR(125), CB(6Ø)                                   B   5Ø
C                                                                         B   6Ø
C **** SET COLUMN AND ROW HEADINGS                                        B   7Ø
C                                                                         B   8Ø
      DO 1Ø1 NV=1,NVAR                                                    B   9Ø
         JCOL(NV,1)=2                                                     B  1ØØ
  1Ø1 JCOL(NV,2)=NV                                                       B  11Ø
      DO 1Ø2 NR=1,NRCON                                                   B  12Ø
```

```
            JROW(NR,1)=1                                                    B  13Ø
            JROW(NR,2)=NR                                                   B  14Ø
            NAR=NVAR+NR                                                     B  15Ø
            JCOL(NAR,1)=1                                                   B  16Ø
  1Ø2 JCOL(NAR,2)=NR                                                        B  17Ø
C                                                                           B  18Ø
C **** READ IN COEFFICIENTS AND RHS OF REAL CONSTRAINTS                     B  19Ø
C                                                                           B  2ØØ
      DO 1Ø3 NR=1,NRCON                                                     B  21Ø
         READ (5,118) TB(NR),(TE(NR,NV),NV=1,NVAR)                          B  22Ø
  1Ø3 CONTINUE                                                              B  23Ø
C                                                                           B  24Ø
C **** PUT IDENTITY MATRIX IN FOR ARTIFICIAL VARIABLES                      B  25Ø
C                                                                           B  26Ø
      DO 1Ø4 NR=1,NRCON                                                     B  27Ø
         NAR=NVAR+NR                                                        B  28Ø
  1Ø4 TE(NR,NAR)=1.                                                         B  29Ø
C                                                                           B  3ØØ
C **** SET C(J)=Ø FOR ALL DECISION VARIABLES AND C(J)=1 FOR ALL             B  31Ø
C **** ARTIFICIAL VARIABLES                                                 B  32Ø
C                                                                           B  33Ø
      DO 1Ø5 NV=1,NVAR                                                      B  34Ø
  1Ø5 C(NV)=Ø.                                                              B  35Ø
      DO 1Ø6 NR=1,NRCON                                                     B  36Ø
         CB(NR)=1.                                                          B  37Ø
         NAR=NVAR+NR                                                        B  38Ø
  1Ø6 C(NAR)=1.                                                             B  39Ø
C                                                                           B  4ØØ
C **** CALCULATE RELATIVE COST COEFFICIENTS CR(.)                           B  41Ø
C                                                                           B  42Ø
      NCOL=NVAR+NRCON                                                       B  43Ø
  1Ø7 DO 1Ø8 NV=1,NCOL                                                      B  44Ø
         CR(NV)=C(NV)                                                       B  45Ø
      DO 1Ø8 NR=1,NRCON                                                     B  46Ø
  1Ø8 CR(NV)=CR(NV)-CB(NR)*TE(NR,NV)                                        B  47Ø
C                                                                           B  48Ø
C **** CHECK FOR OPTIMALITY                                                 B  49Ø
C                                                                           B  5ØØ
      VEVC=Ø.                                                               B  51Ø
      NEVC=Ø                                                                B  52Ø
      DO 1Ø9 NCO=1,NCOL                                                     B  53Ø
         NV=NCO                                                             B  54Ø
         IF (CR(NV).GE.Ø.) GO TO 1Ø9                                        B  55Ø
         IF (CR(NV).GE.VEVC) GO TO 1Ø9                                      B  56Ø
         VEVC=CR(NV)                                                        B  57Ø
         NEVC=NV                                                            B  58Ø
  1Ø9 CONTINUE                                                              B  59Ø
C                                                                           B  6ØØ
C **** IF NEVC=Ø, PHASE 1 IS OPTIMIZED.  CALCULATE OBJECTIVE FUNCTION.      B  61Ø
C                                                                           B  62Ø
      IF (NEVC.EQ.Ø) GO TO 115                                              B  63Ø
C                                                                           B  64Ø
C **** DETERMINE DEPARTING VARIABLES ROW.                                   B  65Ø
C                                                                           B  66Ø
      NDVR=Ø                                                                B  67Ø
      VDVR=1Ø.ØE+2Ø                                                         B  68Ø
      DO 111 NRI=1,NRCON                                                    B  69Ø
         NR=NRI                                                             B  7ØØ
         IF (TE(NR,NEVC).LE.Ø.) GO TO 111                                   B  71Ø
         V=TB(NR)/TE(NR,NEVC)                                               B  72Ø
         IF (NDVR.EQ.Ø) GO TO 11Ø                                           B  73Ø
         IF (V-VDVR) 11Ø,11Ø,111                                            B  74Ø
  11Ø    VDVR=V                                                             B  75Ø
         NDVR=NR                                                            B  76Ø
  111 CONTINUE                                                              B  77Ø
C                                                                           B  78Ø
C **** IF NDVR=Ø, MINIMUM RATIO RULE FAILED.  RETURN.                       B  79Ø
C                                                                           B  8ØØ
      IF (NDVR.EQ.Ø) RETURN                                                 B  81Ø
C                                                                           B  82Ø
C **** PERFORM THE PIVOT.  REPLACE HEADINGS AND COST COEFFICIENT.           B  83Ø
C                                                                           B  84Ø
      JROW(NDVR,1)=JCOL(NEVC,1)                                             B  85Ø
      JROW(NDVR,2)=JCOL(NEVC,2)                                             B  86Ø
      CB(NDVR)=C(NEVC)                                                      B  87Ø
C                                                                           B  88Ø
```

```
C **** COMPUTE NEW TE ARRAY                                              B  89Ø
C                                                                        B  9ØØ
      PIV=TE(NDVR,NEVC)                                                  B  91Ø
      PIB=TB(NDVR)                                                       B  92Ø
      DO 113 NR=1,NRCON                                                  B  93Ø
         IF (NR.EQ.NDVR) GO TO 113                                       B  94Ø
         IF (ABS(TE(NR,NEVC)).LE.Ø.ØØØ5) GO TO 113                       B  95Ø
         PIX=TE(NR,NEVC)/PIV                                             B  96Ø
         TB(NR)=FIX(TB(NR)-PIX*PIB)                                      B  97Ø
         DO 112 NV=1,NCOL                                                B  98Ø
  112    TE(NR,NV)=FIX(TE(NR,NV)-TE(NDVR,NV)*PIX)                        B  99Ø
  113 CONTINUE                                                           B 1ØØØ
      TB(NDVR)=FIX(PIB/PIV)                                              B 1Ø1Ø
      DO 114 NV=1,NCOL                                                   B 1Ø2Ø
  114 TE(NDVR,NV)=FIX(TE(NDVR,NV)/PIV)                                   B 1Ø3Ø
C                                                                        B 1Ø4Ø
C **** END OF PIVOT OPERATIONS.  PROCEED TO NEXT ITERATION.              B 1Ø5Ø
C                                                                        B 1Ø6Ø
      GO TO 1Ø7                                                          B 1Ø7Ø
C                                                                        B 1Ø8Ø
C **** CALCULATE W, THE PHASE-1 OBJECTIVE FUNCTION.
C                                                                        B 11ØØ
  115 W=Ø.                                                               B 111Ø
      DO 116 NR=1,NRCON                                                  B 112Ø
  116 W=W+TB(NR)*CB(NR)                                                  B 113Ø
C                                                                        B 114Ø
C **** INITIALIZE THOSE PORTIONS OF THE TABLEAU ASSIGNED TO THE          B 115Ø
C **** ARTIFICIAL VARIABLES.                                             B 116Ø
C                                                                        B 117Ø
      DO 117 NR=1,NRCON                                                  B 118Ø
      DO 117 NV=1,NCOL                                                   B 119Ø
         IF (NV.LE.NVAR) GO TO 117                                       B 12ØØ
         TE(NR,NV)=Ø.                                                    B 121Ø
  117 CONTINUE                                                           B 122Ø
C                                                                        B 123Ø
C **** UPDATE NCOLI AND NROWI PARAMETERS.                                B 124Ø
C                                                                        B 125Ø
      NROWI=NRCON                                                        B 126Ø
      NCOLI=NVAR                                                         B 127Ø
      RETURN                                                             B  128
C                                                                        B 129Ø
  118 FORMAT (8F1Ø.Ø)                                                    B 13ØØ
C                                                                        B 131Ø
      END                                                                B 132Ø


      SUBROUTINE READ1                                                   C   1Ø
C
C **** SUBROUTINE READ1 READS IN THE GOAL CONSTRAINTS AND OBJECTIVE
C **** FUNCTION TERMS ASSIGNED TO PRIORITY ONE.
C **** SUBROUTINE READ1 IS NOT USED IF REAL CONSTRAINTS ARE PRESENT.
C
      COMMON TT(1Ø,125),TB(6Ø),TE(6Ø,125),TL(6Ø,1Ø),TA(1Ø),TI(1Ø,125),JC  C   2Ø
     1OL(125,2),NCOLI,NROWI,NPRIC,NC(1Ø),JROW(6Ø,2),NVAR,NPRIT,IND(125)   C   3Ø
      COMMON /CHNG/ NCON(6Ø,1Ø),NTOF(1Ø)                                 C   4Ø
C                                                                        C   5Ø
C **** SET COLUMN AND ROW HEADINGS.                                      C   6Ø
C                                                                        C   7Ø
      DO 1Ø1 NV=1,NVAR                                                   C   8Ø
         JCOL(NV,1)=2                                                    C   9Ø
  1Ø1 JCOL(NV,2)=NV                                                      C  1ØØ
      NC11=NC(1)
      DO 1Ø2 NCR=1,NC11
         NC1=NVAR+2*NCR-1                                                C  12Ø
         NC2=NVAR+2*NCR                                                  C  13Ø
         JCOL(NC1,1)=4                                                   C  14Ø
         JCOL(NC1,2)=NCON(NCR,1)                                         C  15Ø
         JCOL(NC2,1)=3                                                   C  16Ø
         JCOL(NC2,2)=NCON(NCR,1)                                         C  17Ø
         JROW(NCR,1)=4                                                   C  18Ø
  1Ø2 JROW(NCR,2)=NCON(NCR,1)                                            C  19Ø
C                                                                        C  2ØØ
C **** READ IN THE GOAL CONSTRAINTS ASSIGNED TO PRIORITY 1.              C  21Ø
C                                                                        C  22Ø
      NC1=NC(1)
      DO 1Ø3 NCR=1,NC1
```

```
         NV1=NVAR+2*NCR-1                                                  C  24Ø
         NV2=NVAR+2*NCR                                                    C  25Ø
         READ (5,1Ø5) TB(NCR),(TE(NCR,NV),NV=1,NVAR)                       C  26Ø
C                                                                          C  27Ø
C **** PUT +1 IN FOR D- AND -1 IN FOR D+.                                  C  28Ø
C                                                                          C  29Ø
         TE(NCR,NV1)=1.                                                    C  3ØØ
         TE(NCR,NV2)=-1.                                                   C  31Ø
  1Ø3 CONTINUE                                                             C  32Ø
      NCOLI=NV2                                                            C  33Ø
      NROWI=NC(1)                                                          C  34Ø
C                                                                          C  35Ø
C **** READ IN THE OBJECTIVE FUNCTION TERMS FOR PRIORITY 1.                C  36Ø
C                                                                          C  37Ø
      NT1=NTOF(1)
      DO 1Ø4 NT=1,NT1
         READ (5,1Ø6) ISUB,ITYPE,WGHT                                      C  39Ø
         CALL PLACE (ISUB,ITYPE,WGHT)                                      C  4ØØ
  1Ø4 CONTINUE                                                             C  41Ø
      RETURN                                                               C  42Ø
C                                                                          C  43Ø
  1Ø5 FORMAT (8F1Ø.Ø)                                                      C  44Ø
  1Ø6 FORMAT (2I5,F1Ø.Ø)                                                   C  45Ø
C                                                                          C  46Ø
      END                                                                  C  47Ø


      SUBROUTINE READ2                                                     D   1Ø
C
C **** SUBROUTINE READ2 READS IN THE GOAL CONSTRAINTS AND OBJECTIVE
C **** FUNCTION TERMS ASSIGNED TO PRIORITY NPRIC.
C **** SUBROUTINE READ2 IS ALSO USED TO READ IN THE FIRST PRIORITY GOAL
C **** CONSTRAINTS AND OBJECTIVE FUNCTION TERMS IF REAL CONSTRAINTS ARE
C **** PRESENT.
C
      COMMON TT(1Ø,125),TB(6Ø),TE(6Ø,125),TL(6Ø,1Ø),TA(1Ø),TI(1Ø,125),JC  D   2Ø
     1OL(125,2),NCOLI,NROWI,NPRIC,NC(1Ø),JROW(6Ø,2),NVAR,NPRIT,IND(125)   D   3Ø
      COMMON /CHNG/ NCON(6Ø,1Ø),NTOF(1Ø)                                   D   4Ø
      IF (NC(NPRIC).EQ.Ø) GO TO 1Ø7                                        D   5Ø
C                                                                          D   6Ø
C **** READ IN THE COEFFICIENTS OF THE X'S.                                D   7Ø
C                                                                          D   8Ø
      NCTMP=NC(NPRIC)
      DO 1Ø6 NRI=1,NCTMP
         NR=NRI+NROWI                                                      D  1ØØ
         NC1=NCOLI+2*NRI-1                                                 D  11Ø
         NC2=NCOLI+2*NRI                                                   D  12Ø
         JCOL(NC1,1)=4                                                     D  13Ø
         JCOL(NC1,2)=NCON(NRI,NPRIC)                                       D  14Ø
         JCOL(NC2,1)=3                                                     D  15Ø
         JCOL(NC2,2)=NCON(NRI,NPRIC)                                       D  16Ø
         READ (5,1Ø9) TB(NR),(TE(NR,NV),NV=1,NVAR)                         D  17Ø
         TE(NR,NC1)=1.                                                     D  18Ø
         TE(NR,NC2)=-1.                                                    D  19Ø
C                                                                          D  2ØØ
C **** PERFORM THE ROW REDUCTION.                                          D  21Ø
C                                                                          D  22Ø
         DO 1Ø2 NRC=1,NROWI                                                D  23Ø
            IF (JROW(NRC,1).NE.2) GO TO 1Ø2                                D  24Ø
            J=JROW(NRC,2)                                                  D  25Ø
            TB(NR)=TB(NR)-TE(NR,J)*TB(NRC)                                 D  26Ø
            DO 1Ø1 NCR=1,NC2                                               D  27Ø
               IF (NCR.EQ.J) GO TO 1Ø1                                     D  28Ø
               TE(NR,NCR)=TE(NR,NCR)-TE(NR,J)*TE(NRC,NCR)                  D  29Ø
  1Ø1       CONTINUE                                                       D  3ØØ
            TE(NR,J)=Ø.                                                    D  31Ø
  1Ø2    CONTINUE                                                          D  32Ø
C                                                                          D  33Ø
C **** DETERMINE THE DEVIATIONAL VARIABLE TO ENTER THE BASIS.              D  34Ø
C                                                                          D  35Ø
         IF (TB(NR)) 1Ø3,1Ø5,1Ø5                                           D  36Ø
C                                                                          D  37Ø
C **** SINCE TB IS LESS THAN ZERO, MULTIPLY THE ROW BY -1 AND ENTER D+     D  38Ø
C **** IN THE BASIS.                                                       D  39Ø
C                                                                          D  4ØØ
  1Ø3    DO 1Ø4 NCR=1,NC2                                                  D  41Ø
```

```
  1Ø4     TE(NR,NCR)=-TE(NR,NCR)                                          D  42Ø
          TB(NR)=-TB(NR)                                                  D  43Ø
          JROW(NR,1)=3                                                    D  44Ø
          JROW(NR,2)=NCON(NRI,NPRIC)                                      D  45Ø
          GO TO 1Ø6                                                       D  46Ø
C                                                                         D  47Ø
C **** SINCE TB IS GREATER THAN OR EQUAL TO ZERO ENTER D- IN THE BASIS.    D  48Ø
C                                                                         D  49Ø
  1Ø5     JROW(NR,1)=4                                                    D  5ØØ
          JROW(NR,2)=NCON(NRI,NPRIC)                                      D  51Ø
  1Ø6 CONTINUE                                                            D  52Ø
C                                                                         D  53Ø
C **** INCREASE THE PARAMETERS NCOLI AND NROWI.                           D  54Ø
C                                                                         D  55Ø
      NCOLI=NC2                                                           D  56Ø
      NROWI=NR                                                            D  57Ø
C                                                                         D  58Ø
C **** READ IN THE OBJECTIVE FUNCTION TERMS FOR PRIORITY NPRIC.           D  59Ø
C                                                                         D  6ØØ
  1Ø7 NTTMP=NTOF(NPRIC)
      DO 1Ø8 NT=1,NTTMP
         READ (5,11Ø) ISUB,ITYPE,WGHT                                     D  62Ø
         CALL PLACE (ISUB,ITYPE,WGHT)                                     D  63Ø
  1Ø8 CONTINUE                                                            D  64Ø
      RETURN                                                               D 65Ø
C                                                                         D  66Ø
  1Ø9 FORMAT (8F1Ø.Ø)                                                     D  67Ø
  11Ø FORMAT (2I5,F1Ø.Ø)                                                  D  68Ø
C                                                                         D  69Ø
      END                                                                 D  7ØØ


      SUBROUTINE PLACE (ISUB,ITYPE,WGHT)                                  E   1Ø
C
C **** SUBROUTINE PLACE PUTS THE OBJECTIVE FUNCTION WEIGHTS FOR THE
C **** DEVIATION VARIABLES AT THE CURRENT PRIORITY LEVEL (NPRIC) IN THE
C **** CORRECT POSITIONS IN THE AUGMENTED TABLEAU.
C ****
C **** ISUB=THE SUBSCRIPT OF THE DEVIATIONAL VARIABLE
C ****
C **** ITYPE=3, IF POSITIVE DEVIATIONAL VARIABLE (D+)
C ****       4, IF NEGATIVE DEVIATIONAL VARIABLE (D-)
C ****
C **** WGHT=THE CARDINAL WEIGHT OF THIS DEVIATIONAL VARIABLE AT THE
C ****      CURRENT PRIORITY LEVEL
C
      COMMON TT(1Ø,125),TB(6Ø),TE(6Ø,125),TL(6Ø,1Ø),TA(1Ø),TI(1Ø,125),JC   E   2Ø
     1OL(125,2),NCOLI,NROWI,NPRIC,NC(1Ø),JROW(6Ø,2),NVAR,NPRIT,IND(125)    E   3Ø
      COMMON /CHNG/ NCON(6Ø,1Ø),NTOF(1Ø)                                  E   4Ø
C                                                                         E   5Ø
C **** PLACE THE WEIGHT IN THE PROPER COLUMN IN THE TOP STUB.             E   6Ø
C                                                                         E   7Ø
      NC1=NVAR+1                                                          E   8Ø
      DO 1Ø1 NCR=NC1,NCOLI                                                E   9Ø
         IF (JCOL(NCR,1).EQ.ITYPE.AND.JCOL(NCR,2).EQ.ISUB) GO TO 1Ø2      E  1ØØ
  1Ø1 CONTINUE                                                            E  11Ø
  1Ø2 TT(NPRIC,NCR)=WGHT                                                  E  12Ø
C                                                                         E  13Ø
C **** PLACE THE WEIGHT IN THE PROPER ROW IN THE LEFT STUB.               E  14Ø
C                                                                         E  15Ø
      DO 1Ø3 NR=1,NROWI                                                   E  16Ø
         IF (JROW(NR,1).EQ.ITYPE.AND.JROW(NR,2).EQ.ISUB) GO TO 1Ø4        E  17Ø
  1Ø3 CONTINUE                                                            E  18Ø
      GO TO 1Ø5                                                           E  19Ø
  1Ø4 TL(NR,NPRIC)=WGHT                                                   E  2ØØ
  1Ø5 CONTINUE                                                            E  21Ø
      RETURN                                                               E 22Ø
C                                                                         E  23Ø
      END                                                                 E  24Ø


      SUBROUTINE CINDX                                                    F   1Ø
C
C **** SUBROUTINE CINDX COMPUTES THE RELATIVE COST COEFFICIENTS FOR EACH
C **** VARIABLE IN THE CURRENT TABLEAU(THE TI( . , . ) ARRAY) AND THE
C **** OBJECTIVE FUNCTION VALUE(THE TA(.) ARRAY) AT THE CURRENT
```

```
C **** PRIORITY(NPRIC)
C
      COMMON TT(1Ø,125),TB(6Ø),TE(6Ø,125),TL(6Ø,1Ø),TA(1Ø),TI(1Ø,125),JC  F   2Ø
     1OL(125,2),NCOLI,NROWI,NPRIC,NC(1Ø),JROW(6Ø,2),NVAR,NPRIT,IND(125)   F   3Ø
C                                                                         F   4Ø
C **** COMPUTE TA(NPRIC) AND TI(NPRIC,NC)   NC=1,....,NCOLI               F   5Ø
C                                                                         F   6Ø
      TA(NPRIC)=Ø.                                                        F   7Ø
      DO 1Ø1 NR=1,NROWI                                                   F   8Ø
  1Ø1 TA(NPRIC)=TA(NPRIC)+TB(NR)*TL(NR,NPRIC)                             F   9Ø
      DO 1Ø2 NCR=1,NCOLI                                                  F  1ØØ
         TI(NPRIC,NCR)=TT(NPRIC,NCR)                                      F  11Ø
      DO 1Ø2 NR=1,NROWI                                                   F  12Ø
  1Ø2 TI(NPRIC,NCR)=TI(NPRIC,NCR)-TE(NR,NCR)*TL(NR,NPRIC)                 F  13Ø
      RETURN                                                               F 14Ø
C                                                                         F  15Ø
      END                                                                 F  16Ø


      SUBROUTINE TEST (NEVC,NDVR)                                         G   1Ø
C
C **** SUBROUTINE TEST DETERMINES THE NEXT ENTERING VARIABLE'S COLUMN
C **** (NEVC) AND THE NEXT DEPARTING VARIABLE'S ROW(NDVR).  IF NO
C **** FURTHER OPTIMIZATION IS POSSIBLE, THE VALUE NEVC=Ø IS RETURNED.
C **** IF NDVR=Ø IS RETURNED, NO MINIMUM POSITIVE RATIO COULD BE FOUND
C **** IN THE CURRENT PIVOT OPERATION,I.E., ALL OF THE COEFFICIENTS
C **** TE( . ,NEVC) ARE NONPOSITIVE.
C
      COMMON TT(1Ø,125),TB(6Ø),TE(6Ø,125),TL(6Ø,1Ø),TA(1Ø),TI(1Ø,125),JC  G   2Ø
     1OL(125,2),NCOLI,NROWI,NPRIC,NC(1Ø),JROW(6Ø,2),NVAR,NPRIT,IND(125)   G   3Ø
      NDVR=Ø                                                               G  4Ø
      NEVC=Ø                                                               G  5Ø
      VEVC=Ø.                                                             G   6Ø
      VDVR=1Ø.ØE+2Ø                                                       G   7Ø
C                                                                         G   8Ø
C **** DETERMINE ENTERING VARIABLE'S COLUMN.                              G   9Ø
C                                                                         G  1ØØ
      DO 1Ø1 NCR=1,NCOLI                                                  G  11Ø
         IF (TI(NPRIC,NCR).GE.Ø.) GO TO 1Ø1                               G  12Ø
         IF (IND(NCR).EQ.Ø) GO TO 1Ø1                                     G  13Ø
         IF (TI(NPRIC,NCR).GE.VEVC) GO TO 1Ø1                             G  14Ø
         NEVC=NCR                                                         G  15Ø
         VEVC=TI(NPRIC,NCR)                                               G  16Ø
  1Ø1 CONTINUE                                                            G  17Ø
C                                                                         G  18Ø
C **** IF NEVC=Ø, SUBPROBLEM NPRIC IS OPTIMIZED. RETURN.                  G  19Ø
C                                                                         G  2ØØ
      IF (NEVC.EQ.Ø) RETURN                                               G  21Ø
C                                                                         G  22Ø
C **** DETERMINE DEPARTING VARIABLE'S ROW.                                G  23Ø
C                                                                         G  24Ø
      DO 1Ø5 NR=1,NROWI                                                   G  25Ø
         IF (TE(NR,NEVC).LE.Ø.) GO TO 1Ø5                                 G  26Ø
         V=TB(NR)/TE(NR,NEVC)                                             G  27Ø
         IF (NDVR.EQ.Ø) GO TO 1Ø4                                         G  28Ø
         IF (V-VDVR) 1Ø4,1Ø2,1Ø5                                          G  29Ø
  1Ø2    DO 1Ø3 NP=1,NPRIC                                                G  3ØØ
            IF (TL(NR,NP)-TL(NDVR,NP)) 1Ø5,1Ø3,1Ø4                        G  31Ø
  1Ø3    CONTINUE                                                         G  32Ø
  1Ø4    VDVR=V                                                           G  33Ø
         NDVR=NR                                                          G  34Ø
  1Ø5 CONTINUE                                                            G  35Ø
      RETURN                                                              G  36Ø
C                                                                         G  37Ø
      END                                                                 G  38Ø


      SUBROUTINE PERM (NEVC,NDVR)                                         H   1Ø
C
C **** SUBROUTINE PERM PERFORMS THE PIVOT OPERATION USING THE PIVOT
C **** ELEMENT IN COLUMN NEVC AND ROW NDVR AND COMPUTES THE NEW TABLEAU.
C
      COMMON TT(1Ø,125),TB(6Ø),TE(6Ø,125),TL(6Ø,1Ø),TA(1Ø),TI(1Ø,125),JC  H   2Ø
     1OL(125,2),NCOLI,NROWI,NPRIC,NC(1Ø),JROW(6Ø,2),NVAR,NPRIT,IND(125)   H   3Ø
C                                                                         H   4Ø
```

```
C **** REPLACE HEADING FOR ROW NDVR.                                    H   5Ø
C                                                                       H   6Ø
      JROW(NDVR,1)=JCOL(NEVC,1)                                         H   7Ø
      JROW(NDVR,2)=JCOL(NEVC,2)                                         H   8Ø
C                                                                       H   9Ø
C **** REPLACE TL VECTOR FOR ROW NDVR                                   H  1ØØ
C                                                                       H  11Ø
      DO 1Ø1 NP=1,NPRIC                                                 H  12Ø
 1Ø1 TL(NDVR,NP)=TT(NP,NEVC)                                            H  13Ø
C                                                                       H  14Ø
C **** COMPUTE NEW TE ARRAY.                                            H  15Ø
C                                                                       H  16Ø
      PIV=TE(NDVR,NEVC)                                                 H  17Ø
      PIB=TB(NDVR)                                                      H  18Ø
      DO 1Ø3 NR=1,NROWI                                                 H  19Ø
         IF (NR.EQ.NDVR) GO TO 1Ø3                                      H  2ØØ
         IF (ABS(TE(NR,NEVC)).LE.Ø.ØØØ5) GO TO 1Ø3                      H  21Ø
         PIX=TE(NR,NEVC)/PIV                                            H  22Ø
         TB(NR)=FIX(TB(NR)-PIX*PIB)                                     H  23Ø
         DO 1Ø2 NCR=1,NCOLI                                             H  24Ø
 1Ø2     TE(NR,NCR)=FIX(TE(NR,NCR)-TE(NDVR,NCR)*PIX)                    H  25Ø
 1Ø3 CONTINUE                                                           H  26Ø
      TB(NDVR)=FIX(PIB/PIV)                                             H  27Ø
      DO 1Ø4 NCR=1,NCOLI                                                H  28Ø
 1Ø4 TE(NDVR,NCR)=FIX(TE(NDVR,NCR)/PIV)                                 H  29Ø
      RETURN                                                            H  3ØØ
C                                                                       H  31Ø
      END                                                               H  32Ø


      FUNCTION FIX(Z)                                                   I   1Ø
C
C **** FUNCTION FIX BRINGS FLOATING POINT VALUES THAT ARE WITHIN 1.E-3
C **** OF AN INTEGER TO THAT INTEGER.
C
      FIX=AINT(Z+SIGN(.5,Z))
      IF (ABS(FIX-Z).GT. 1.E-3) FIX=Z
      RETURN                                                            I  15Ø
C                                                                       I  16Ø
      END                                                               I  17Ø


      SUBROUTINE POUT                                                   J   1Ø
C
C **** SUBROUTINE POUT PREPARES AND PRINTS THE SOLUTION INFORMATION.
C
      COMMON TT(1Ø,125),TB(6Ø),TE(6Ø,125),TL(6Ø,1Ø),TA(1Ø),TI(1Ø,125),JC  J   2Ø
     1OL(125,2),NCOLI,NROWI,NPRIC,NC(1Ø),JROW(6Ø,2),NVAR,NPRIT,IND(125)  J   3Ø
      COMMON /CHNG/ NCON(6Ø,1Ø),NTOF(1Ø)                                J   4Ø
      DIMENSION WOUT(125,4), RLHS(6Ø,1Ø), WM(6Ø), WP(6Ø)                J   5Ø
      DIMENSION DIFF(6Ø)                                                J   6Ø
      WRITE (6,122)                                                     J   7Ø
      WRITE (6,123) NPRIC,NROWI                                         J   8Ø
C                                                                       J   9Ø
C **** OUTPUT ARRAY IS ZEROED.                                          J  1ØØ
C                                                                       J  11Ø
      DO 1Ø1 I=1,125                                                    J  12Ø
      DO 1Ø1 J=1,4                                                      J  13Ø
 1Ø1 WOUT(I,J)=Ø.                                                       J  14Ø
C                                                                       J  15Ø
C **** OUTPUT ARRAY IS FILLED.                                          J  16Ø
C                                                                       J  17Ø
      DO 1Ø2 NP=1,NPRIC                                                 J  18Ø
 1Ø2 WOUT(NP,1)=FIX(TA(NP))                                             J  19Ø
      DO 1Ø3 NR=1,NROWI                                                 J  2ØØ
         I1=JROW(NR,1)                                                  J  21Ø
         I2=JROW(NR,2)                                                  J  22Ø
 1Ø3 WOUT(I2,I1)=FIX(TB(NR))                                            J  23Ø
C                                                                       J  24Ø
C **** IF ALL PRIORITIES HAVE BEEN INCLUDED, PRINT OPTIMAL SOLUTION.    J  25Ø
C **** IF NOT, WE MUST CALCULATE VALUES FOR REMAINING TA'S AND D- AND D+ J  26Ø
C                                                                       J  27Ø
      IF (NPRIC.GE.NPRIT) GO TO 114                                     J  28Ø
      NP1=NPRIC+1                                                       J  29Ø
      DO 113 NP=NP1,NPRIT                                               J  3ØØ
```

```
            TA(NP)=Ø.                                                    J  31Ø
            IF (NC(NP).EQ.Ø) GO TO 1Ø6                                   J  32Ø
C                                                                        J  33Ø
C **** READ IN THE GOAL CONSTRAINTS ASSIGNED TO PRIORITY NP.             J  34Ø
C                                                                        J  35Ø
            NCTMP=NC(NP)
            DO 1Ø5 NCI=1,NCTMP
               NR=NROWI+NCI                                              J  37Ø
               READ (5,124) TB(NR),(TE(NR,NV),NV=1,NVAR)                 J  38Ø
               RLHS(NCI,NP)=Ø.                                           J  39Ø
               DO 1Ø4 NV=1,NVAR                                          J  4ØØ
  1Ø4          RLHS(NCI,NP)=RLHS(NCI,NP)+TE(NR,NV)*WOUT(NV,2)            J  41Ø
               DIFF(NCI)=TB(NR)-RLHS(NCI,NP)                             J  42Ø
  1Ø5       CONTINUE                                                     J  43Ø
C                                                                        J  44Ø
C **** READ THE OBJECTIVE FUNCTION TERMS FOR PRIORITY NP.                J  45Ø
C                                                                        J  46Ø
  1Ø6       NTTMP=NTOF(NP)
            DO 112 NT=1,NTTMP
               READ (5,125) ISUB,ITYPE,WGHT                              J  48Ø
               IF (NC(NP).EQ.Ø) GO TO 111                                J  49Ø
               NCTMP=NC(NP)
               DO 11Ø NCI=1,NCTMP
                  IF (ISUB.NE.NCON(NCI,NP)) GO TO 11Ø                    J  51Ø
                  IF (DIFF(NCI)) 1Ø7,1Ø8,1Ø9                             J  52Ø
  1Ø7             IF (ITYPE.NE.3) GO TO 11Ø                              J  53Ø
                  WOUT(ISUB,3)=-DIFF(NCI)                                J  54Ø
  1Ø8             GO TO 11Ø                                              J  55Ø
  1Ø9             IF (ITYPE.NE.4) GO TO 11Ø                              J  56Ø
                  WOUT(ISUB,4)=DIFF(NCI)                                 J  57Ø
  11Ø          CONTINUE                                                  J  58Ø
  111          TA(NP)=TA(NP)+WGHT*WOUT(ISUB,ITYPE)                       J  59Ø
  112       CONTINUE                                                     J  6ØØ
            NROWI=NROWI+NC(NP)                                           J  61Ø
C                                                                        J  62Ø
C **** FILL IN THE OUTPUT VALUE FOR ATTAINMENT OF PRIORITY NP.           J  63Ø
C                                                                        J  64Ø
            WOUT(NP,1)=FIX(TA(NP))                                       J  65Ø
  113    CONTINUE                                                        J  66Ø
C                                                                        J  67Ø
C **** PRINT OPTIMAL SOLUTION                                            J  68Ø
C                                                                        J  69Ø
  114    WRITE (6,126)                                                   J  7ØØ
         WRITE (6,127)                                                   J  71Ø
         DO 115 NV=1,NVAR                                                J  72Ø
            WRITE (6,128) NV,WOUT(NV,2)                                  J  73Ø
  115    CONTINUE                                                        J  74Ø
         WRITE (6,126)                                                   J  75Ø
         WRITE (6,129)                                                   J  76Ø
         DO 116 NP=1,NPRIT                                               J  77Ø
            IF (NC(NP).EQ.Ø) GO TO 116                                   J  78Ø
            NCTMP=NC(NP)
         DO 139 NCO=1,NCTMP
            N=NCON(NCO,NP)                                               J  8ØØ
            WRITE (6,13Ø) NP,N,WOUT(N,3),WOUT(N,4)                       J  81Ø
  139    CONTINUE
  116    CONTINUE                                                        J  82Ø
         WRITE (6,126)                                                   J  83Ø
         WRITE (6,131)                                                   J  84Ø
         DO 117 NP=1,NPRIT                                               J  85Ø
            WRITE (6,132) NP,WOUT(NP,1)                                  J  86Ø
  117    CONTINUE                                                        J  87Ø
         WRITE (6,126)                                                   J  88Ø
         WRITE (6,133)                                                   J  89Ø
         WRITE (6,134)                                                   J  9ØØ
         I=MAXØ(NPRIT,NVAR,NROWI)                                        J  91Ø
         DO 121 K=1,I                                                    J  92Ø
            IF (K.GT.NPRIT) GO TO 119                                    J  93Ø
            IF (K.GT.NVAR) GO TO 118                                     J  94Ø
            WRITE (6,135) K,(WOUT(K,J),J=1,4)                            J  95Ø
            GO TO 121                                                    J  96Ø
  118       WRITE (6,136) K,WOUT(K,1),(WOUT(K,J),J=3,4)                  J  97Ø
            GO TO 121                                                    J  98Ø
  119       IF (K.GT.NVAR) GO TO 12Ø                                     J  99Ø
            WRITE (6,137) K,(WOUT(K,J),J=2,4)                            J 1ØØØ
            GO TO 121                                                    J 1Ø1Ø
```

```
  12Ø    WRITE (6,138) K,(WOUT(K,J),J=3,4)                                J 1Ø2Ø
  121 CONTINUE                                                            J 1Ø3Ø
      WRITE (6,126)                                                       J 1Ø4Ø
      RETURN                                                               J 1Ø5
C                                                                         J 1Ø6Ø
  122 FORMAT (1H1)                                                        J 1Ø7Ø
  123 FORMAT (/  39H  THE OPTIMIZATION ENDED ON SUBPROBLEM ,I5 /  13H  T
     1HERE WERE ,I5, 42H CONSTRAINTS IN THE FINAL OPTIMAL TABLEAU.)       J 1Ø9Ø
  124 FORMAT (8F1Ø.Ø)                                                     J 11ØØ
  125 FORMAT (2I5,F1Ø.Ø)                                                  J 111Ø
  126 FORMAT (//12Ø(1H*))                                                 J 112Ø
  127 FORMAT (1HØ, 52HTHE OPTIMAL SOLUTION FOR THE DECISION VARIABLES X(  J 113Ø
     1J))                                                                 J 114Ø
  128 FORMAT (1HØ,  2HX(,I3,  2H)=,F15.4)                                 J 115Ø
  129 FORMAT (1HØ, 25HTHE GOAL ACHIEVEMENTS ARE //   9H PRIORITY,2X, 11H
     1GOAL NUMBER,2X, 16HOVER-ACHIEVEMENT,2X, 17HUNDER-ACHIEVEMENT)       J 117Ø
  13Ø FORMAT (4X,I2,1ØX,I2,1ØX,F1Ø.4,1ØX,F1Ø.4)                           J 118Ø
  131 FORMAT (1HØ, 29HTHE PRIORITY ACHIEVEMENTS ARE //   9H PRIORITY,8X,
     1 11HACHIEVEMENT)                                                    J 12ØØ
  132 FORMAT (4X,I2,1ØX,F1Ø.4)                                            J 121Ø
  133 FORMAT (1HØ, 15H OUTPUT SUMMARY)                                    J 122Ø
  134 FORMAT (1HØ,  9HSUBSCRIPT,11X,  8H   A OPT,7X,  8H   X OPT,7X,  9H  J 123Ø
     1  POS DEV,6X,  9H  NEG DEV /)
  135 FORMAT (I8,7X,4F15.4)                                               J 125Ø
  136 FORMAT (I8,7X,F15.4,15X,2F15.4)                                     J 126Ø
  137 FORMAT (I8,22X,3F15.4)                                              J 127Ø
  138 FORMAT (I8,37X,2F15.4)                                              J 128Ø
C                                                                         J 129Ø
      END                                                                 J 13ØØ
```

ALGORITHM 558

A Program for the Multifacility Location Problem with Rectilinear Distance by the Minimum-Cut Approach [H]

TO-YAT CHEUNG
University of Ottawa

Key Words and Phrases: multifacility, optimal location, rectilinear distance, minimum cut
CR Categories: 3.57, 5.41
Language: Fortran

DESCRIPTION

This algorithm complements [1], where more details are given. The coding of the algorithm consists of two subroutines, LOCATE and NETFLO. A user's program calls LOCATE, which contains the major coding, and calls NETFLO for solving the minimum-cut network flow subproblems.

ACKNOWLEDGMENTS

This author wishes to thank Prof. A. Nijenhuis, Prof. H. Wilf, and Academic Press for their permission to use and modify their subroutine NETFLO. The referees' efforts in testing the program are very much appreciated.

REFERENCE

1. CHEUNG, T.Y. Multifacility location problem with rectilinear distance by the minimum cut approach. *ACM Trans. Math. Softw. 6,* 3 (Sept. 1980), 387–390.

ALGORITHM

```
C                                                                    TP1 ØØØØ
C *TEST PROGRAM #1* (DATA INPUT FROM CARDS)                          TP1 ØØØ1
C THIS IS THE FIRST OF TWO PROGRAMS FOR THE TESTING OF THE           TP1 ØØØ2
C SUBROUTINE 'LOCATE'. ALL IT DOES IS TO READ THE DATA NVPT,         TP1 ØØØ3
C NFPT, VWT AND VFWT FROM CARDS, CALL 'LOCATE', AND PRINT THE        TP1 ØØØ4
C SOLUTION POST.                                                     TP1 ØØØ5
C IT FIRST READS AN INTEGER NPRLM FROM A CARD, USING                 TP1 ØØØ6
C FORMAT(I3), WHERE NPRLM IS THE NUMBER OF PROBLEMS TO BE            TP1 ØØØ7
C TESTED IN A SINGLE RUN. THEN, FOR EACH PROBLEM TESTED,             TP1 ØØØ8
C A SET OF 1+2*NVPT DATA CARDS WILL BE READ. THE FORMAT              TP1 ØØØ9
C AND SETUP OF THESE CARDS SHOULD BE CLEAR FROM THE READ             TP1 ØØ1Ø
```

Received 17 February 1978; revised 27 June 1979; accepted 11 December 1979.

This research was supported by the Natural Sciences and Engineering Research Council of Canada under Grant A8963.
Author's address: Department of Computer Science, University of Ottawa, Ottawa, Ont., Canada K1N 6N5.

```
C STATEMANTS AT STATEMENT LABEL 2, IN LOOP 7 AND LOOP 1Ø.                TP1 ØØ11
C                                                                        TP1 ØØ12
      INTEGER VWT(3Ø,3Ø),VFWT(3Ø,6Ø),POST(3Ø)                            TP1 ØØ13
C                                                                        TP1 ØØ14
C FOR EXPLANATION OF THESE VECTORS, SEE SUBROUTINE 'LOCATE'.             TP1 ØØ15
C                                                                        TP1 ØØ16
      READ(5,1) NPRLM                                                    TP1 ØØ17
    1 FORMAT(I3)                                                         TP1 ØØ18
      NP=1                                                               TP1 ØØ19
    2 READ(5,3) NVPT,NFPT                                                TP1 ØØ2Ø
    3 FORMAT(2I3)                                                        TP1 ØØ21
      WRITE(6,4) NP,NVPT,NFPT                                            TP1 ØØ22
    4 FORMAT(1H1/////14H TEST PROBLEM ,I3//11H NUMBER OF ,               TP1 ØØ23
     119HVARIABLE POINTS IS ,I3,24H;    NUMBER OF FIXED POI,             TP1 ØØ24
     17HNTS IS ,I2)                                                      TP1 ØØ25
      WRITE(6,5)                                                         TP1 ØØ26
    5 FORMAT(2ØHØWEIGHT MATRIX VWT :/)                                   TP1 ØØ27
      DO 7 I=1,NVPT                                                      TP1 ØØ28
        READ(5,6) (VWT(I,J),J=1,NVPT)                                    TP1 ØØ29
    6   FORMAT(25I3)                                                     TP1 ØØ3Ø
    7 WRITE(6,8) (VWT(I,J),J=1,NVPT)                                     TP1 ØØ31
    8 FORMAT(5X,22(I3,1X)/13X,2Ø(I3,1X)/13X,2Ø(I3,1X))                   TP1 ØØ32
      WRITE(6,9)                                                         TP1 ØØ33
    9 FORMAT(21HØWEIGHT MATRIX VFWT :/)                                  TP1 ØØ34
      DO 1Ø I=1,NVPT                                                     TP1 ØØ35
        READ(5,6) (VFWT(I,J),J=1,NFPT)                                   TP1 ØØ36
   1Ø WRITE(6,8) (VFWT(I,J),J=1,NFPT)                                    TP1 ØØ37
      CALL LOCATE(NVPT,NFPT,VWT,VFWT,POST)                               TP1 ØØ38
      WRITE(6,11) (POST(I),I=1,NVPT)                                     TP1 ØØ39
   11 FORMAT(38H-OPTIMAL LOCATIONS OF VARIABLE POINTS ,                  TP1 ØØ4Ø
     142HCOINCIDE WITH THE FOLLOWING FIXED POINTS :/3X,                  TP1 ØØ41
     122(I3,1X)/3X,2Ø(I3,1X))                                            TP1 ØØ42
C                                                                        TP1 ØØ43
C CHECK WHETHER THERE IS MORE TEST PROBLEM                               TP1 ØØ44
      NP=NP+1                                                            TP1 ØØ45
      IF(NP.LE.NPRLM) GO TO 2                                            TP1 ØØ46
      STOP                                                               TP1 ØØ47
      END                                                                TP1 ØØ48
```

```
C                                                                        TP2 ØØØØ
C *TEST PROGRAM #2* (DATA GENERATED RANDOMLY)                            TP2 ØØØ1
C THIS IS THE SECOND OF THE TWO MAIN PROGRAMS FOR TESTING                TP2 ØØØ2
C THE SUBROUTINE 'LOCATE'.  IT TESTS DATA GENERATED RANDOMLY             TP2 ØØØ3
C BY A SUBROUTINE CALLED RANDU. IT FIRST READS AN INTEGER                TP2 ØØØ4
C NPRLM FROM A CARD, USING FORMAT(I3), WHERE NPRLM IS THE                TP2 ØØØ5
C NUMBER OF PROBLEMS TO BE TESTED IN A SINGLE RUN. FOR EACH              TP2 ØØØ6
C PROBLEM TESTED, THE USER PROVIDES ONLY A SINGLE CARD TO BE             TP2 ØØØ7
C READ BY THE FOLLOWING STATEMENTS:                                      TP2 ØØØ8
C                                                                        TP2 ØØØ9
C       2 READ(5,3)NVPT,NFPT,NCHECK,MVWT,MVFWT,RANGE                     TP2 ØØ1Ø
C                                                                        TP2 ØØ11
C       3 FORMAT(5I3,F6.2)                                               TP2 ØØ12
C                                                                        TP2 ØØ13
C WHERE                                                                  TP2 ØØ14
C NVPT   - NUMBER OF VARIABLE POINTS.                                    TP2 ØØ15
C NFPT   - NUMBER OF FIXED POINTS.                                       TP2 ØØ16
C NCHECK - NUMBER OF SETS OF VARIABLE-POINT LOCATIONS                    TP2 ØØ17
C          GENERATED FOR CHECKING THE OPTIMALITY OF THE                  TP2 ØØ18
C          OBTAINED SOLUTION.                                            TP2 ØØ19
C MVWT   - UPPER BOUND ON THE WEIGHTS (F1ØØØ) BETWEEN THE                TP2 ØØ2Ø
C          VARIABLE POINTS.                                              TP2 ØØ21
C MVFWT  - UPPER BOUND ON THE WEIGHTS (F1ØØØ) BETWEEN THE                TP2 ØØ22
C          FIXED AND VARIABLE POINTS.                                    TP2 ØØ23
C RANGE  - UPPER BOUND ON THE RANGE OF LOCATIONS GENERATED.              TP2 ØØ24
C                                                                        TP2 ØØ25
C USING THESE INPUT DATA, THIS PROGRAM FIRST RANDOMLY GENERATES          TP2 ØØ26
C A SET OF FIXED-POINTS LOCATIONS. IT THEN GENERATES RANDOMLY            TP2 ØØ27
C NCHECK SETS OF VARIABLE-POINT LOCATIONS AND COMPARES THEM              TP2 ØØ28
C WITH THE OPTIMAL LOCATIONS OBTAINED BY 'LOCATE'.                       TP2 ØØ29
C THIS MAIN PROGRAM INCLUDES THREE OTHER SUBROUTINES:                    TP2 ØØ3Ø
C RANDU, CHECK AND SUM.                                                  TP2 ØØ31
C                                                                        TP2 ØØ32
C                                                                        TP2 ØØ33
      INTEGER VWT(3Ø,3Ø),VFWT(3Ø,6Ø),POST(3Ø)                            TP2 ØØ34
```

```
C                                                                       TP2 ØØ35
C FOR EXPLANATION OF THESE VECTORS, SEE SUBROUTINE 'LOCATE'             TP2 ØØ36
C                                                                       TP2 ØØ37
      READ(5,1) NPRLM                                                   TP2 ØØ38
    1 FORMAT(I3)                                                        TP2 ØØ39
      IY=2*NPRLM+1                                                      TP2 ØØ4Ø
      NP=1                                                              TP2 ØØ41
    2 READ(5,3) NVPT,NFPT,NCHECK,MVWT,MVFWT,RANGE                       TP2 ØØ42
    3 FORMAT(5I3,F6.2)                                                  TP2 ØØ43
      WRITE(6,4) NP,NVPT,NFPT                                           TP2 ØØ44
    4 FORMAT(1H1/////14H TEST PROBLEM ,I3//11H NUMBER OF ,              TP2 ØØ45
     119HVARIABLE POINTS IS ,I3,24H;    NUMBER OF FIXED POI,            TP2 ØØ46
     17HNTS IS ,I2)                                                     TP2 ØØ47
      WRITE(6,5) MVWT                                                   TP2 ØØ48
    5 FORMAT(35HØWEIGHT MATRIX VWT (UPPER BOUND IS ,I3,2H):/)           TP2 ØØ49
C                                                                       TP2 ØØ5Ø
C GENERATE MATRIX VWT CONTAINING WEIGHTS BETWEEN VARIABLE               TP2 ØØ51
C POINTS.                                                               TP2 ØØ52
      DO 9 I=1,NVPT                                                     TP2 ØØ53
        DO 6 J=I,NVPT                                                   TP2 ØØ54
          IX=IY                                                         TP2 ØØ55
          CALL RANDU(IX,IY,YFL)                                         TP2 ØØ56
          TEMP=FLOAT(MVWT)*YFL                                          TP2 ØØ57
          VWT(I,J)=INT(TEMP)                                            TP2 ØØ58
          VWT(J,I)=VWT(I,J)                                             TP2 ØØ59
          IF (I.EQ.J) VWT(I,J)=Ø                                        TP2 ØØ6Ø
    6   CONTINUE                                                        TP2 ØØ61
        WRITE(6,7) (VWT(I,J),J=1,NVPT)                                  TP2 ØØ62
    7 FORMAT(5X,22(I3,1X)/13X,2Ø(I3,1X)/13X,2Ø(I3,1X))                  TP2 ØØ63
C                                                                       TP2 ØØ64
C GENERATE MATRIX VFWT CONTAINING WEIGHTS BETWEEN VARIABLE AND          TP2 ØØ65
C FIXED POINTS.                                                         TP2 ØØ66
        DO 8 J=1,NFPT                                                   TP2 ØØ67
          IX=IY                                                         TP2 ØØ68
          CALL RANDU(IX,IY,YFL)                                         TP2 ØØ69
          TEMP=FLOAT(MVFWT)*YFL                                         TP2 ØØ7Ø
          VFWT(I,J)=INT(TEMP)                                           TP2 ØØ71
    8   CONTINUE                                                        TP2 ØØ72
    9 CONTINUE                                                          TP2 ØØ73
      WRITE(6,1Ø) MVFWT                                                 TP2 ØØ74
   1Ø FORMAT(36HØWEIGHT MATRIX VFWT (UPPER BOUND IS ,I3,                TP2 ØØ75
     12H):/)                                                            TP2 ØØ76
      DO 11 I=1,NVPT                                                    TP2 ØØ77
   11 WRITE(6,7) (VFWT(I,J),J=1,NFPT)                                   TP2 ØØ78
C                                                                       TP2 ØØ79
C FIND AN OPTIMAL SOLUTION BY CALLING 'LOCATE'.                         TP2 ØØ8Ø
      CALL LOCATE(NVPT,NFPT,VWT,VFWT,POST)                              TP2 ØØ81
      WRITE(6,12) (POST(I),I=1,NVPT)                                    TP2 ØØ82
   12 FORMAT(38H-OPTIMAL LOCATIONS OF VARIABLE POINTS ,                 TP2 ØØ83
     142HCOINCIDE WITH THE FOLLOWING FIXED POINTS ://3X,                TP2 ØØ84
     122(I3,1X)/3X,2Ø(I3,1X))                                           TP2 ØØ85
C                                                                       TP2 ØØ86
C TO TEST THE OPTIMALTY OF THE SOLUTION OBTAINED BY 'LOCATE',           TP2 ØØ87
C THE SUBROUTINE CHECK GENERATES NCHECK SETS OF VARIABLE-               TP2 ØØ88
C POINT LOCATIONS IN THE RANGE (Ø,RANGE) FOR COMPARISON.                TP2 ØØ89
      CALL CHECK(NVPT,NFPT,VWT,VFWT,POST,NCHECK,RANGE)                  TP2 ØØ9Ø
      NP=NP+1                                                           TP2 ØØ91
      IF(NP.LE.NPRLM) GO TO 2                                           TP2 ØØ92
      STOP                                                              TP2 ØØ93
      END                                                               TP2 ØØ94


      SUBROUTINE RANDU(IX,IY,YFL)                                       TP2 ØØ95
C                                                                       TP2 ØØ96
C THIS GENERATOR GENERATES A NUMBER RANDOMLY AND UNIFORMLY              TP2 Ø097
C IN (Ø,1). SEE MACLAREN AND MARSALIA, J. ACM, VOL. 12,                 TP2 ØØ98
C PP.83-89, AND IBM SCIENTIFIC SUBROUTINE PACKAGE.                      TP2 ØØ99
C                                                                       TP2 Ø1ØØ
      IY=IX*65539                                                       TP2 Ø1Ø1
      IF (IY) 1,2,2                                                     TP2 Ø1Ø2
    1 IY=IY+2147483647+1                                                TP2 Ø1Ø3
    2 YFL=IY                                                            TP2 Ø1Ø4
      YFL=YFL*Ø.4656613E-9                                              TP2 Ø1Ø5
      RETURN                                                            TP2 Ø1Ø6
      END                                                               TP2 Ø1Ø7
```

```
      SUBROUTINE CHECK(NVPT,NFPT,VWT,VFWT,POST,NCHECK,RANGE)          TP2 Ø1Ø8
C                                                                     TP2 Ø1Ø9
C THIS SUBROUTINE FIRST RANDOMLY GENERATES A SET OF FIXED-            TP2 Ø11Ø
C POINT LOCATIONS. IT THEN GENERATES RANDOMLY NCHECK SETS OF          TP2 Ø111
C VARIABLE-POINT LOCATIONS AND COMPARES THEM WITH THE                 TP2 Ø112
C OPTIMAL LOCATIONS OBTAINED BY THE ALGORITHM.                        TP2 Ø113
C                                                                     TP2 Ø114
      INTEGER VWT(3Ø,1),VFWT(3Ø,1),POST(1)                            TP2 Ø115
      DIMENSION VPLOC(3Ø),FPLOC(6Ø)                                   TP2 Ø116
C                                                                     TP2 Ø117
C VPLOC - VARIABLE-POINT LOCATIONS.                                   TP2 Ø118
C FPLOC - FIXED-POINT LOCATIONS.                                      TP2 Ø119
C THE MEANINGS OF THE OTHER VARIABLES ARE GIVEN IN THE TEST           TP2 Ø12Ø
C PROGRAM #2.                                                         TP2 Ø121
C                                                                     TP2 Ø122
C GENERATE FIXED-POINT LOCATIONS.                                     TP2 Ø123
C                                                                     TP2 Ø124
      IY=NCHECK*2+1                                                   TP2 Ø125
      FPLOC(1)=Ø.                                                     TP2 Ø126
      DO 1 I=2,NFPT                                                   TP2 Ø127
        IX=IY                                                         TP2 Ø128
        CALL RANDU(IX,IY,YFL)                                         TP2 Ø129
        FPLOC(I)=YFL*RANGE                                            TP2 Ø13Ø
    1 CONTINUE                                                        TP2 Ø131
      N1=NFPT-1                                                       TP2 Ø132
      DO 3 I=2,N1                                                     TP2 Ø133
        I1=I+1                                                        TP2 Ø134
        DO 2 J=I1,NFPT                                                TP2 Ø135
          IF (FPLOC(I).LE.FPLOC(J)) GO TO 2                           TP2 Ø136
          T=FPLOC(I)                                                  TP2 Ø137
          FPLOC(I)=FPLOC(J)                                           TP2 Ø138
          FPLOC(J)=T                                                  TP2 Ø139
    2   CONTINUE                                                      TP2 Ø14Ø
    3 CONTINUE                                                        TP2 Ø141
      T=FPLOC(NFPT)                                                   TP2 Ø142
      WRITE(6,4) RANGE,T                                              TP2 Ø143
    4 FORMAT(14HØRANDOM CHECK://3X,13HFIXED-POINT   ,                 TP2 Ø144
     15ØH  LOCATIONS ARE GENERATED RANDOMLY IN THE RANGE (Ø,          TP2 Ø145
     14H.ØØ,,F6.2,1H)/3X,29HVARIABLE-POINT LOCATIONS ARE ,            TP2 Ø146
     138HGENERATED RANDOMLY IN THE RANGE (Ø.ØØ,,F6.2,1H)//            TP2 Ø147
     14X,26HFIXED-POINT LOCATIONS ARE:)                               TP2 Ø148
      WRITE(6,5) (FPLOC(I),I=1,NFPT)                                  TP2 Ø149
    5 FORMAT(7X,12F7.2)                                               TP2 Ø15Ø
C                                                                     TP2 Ø151
C GENERATE VARIABLE-POINT LOCATIONS AND EVALUATE THE                  TP2 Ø152
C WEIGHTED SUM OF DISTANCES.                                          TP2 Ø153
      DO 6 I=1,NVPT                                                   TP2 Ø154
        J=POST(I)                                                     TP2 Ø155
        VPLOC(I)=FPLOC(J)                                             TP2 Ø156
    6 CONTINUE                                                        TP2 Ø157
      WRITE(6,7)                                                      TP2 Ø158
    7 FORMAT(42HØ    OPTIMAL VARIABLE-POINT LOCATIONS ARE:)           TP2 Ø159
      WRITE(6,5) (VPLOC(I),I=1,NVPT)                                  TP2 Ø16Ø
      S=SUM(NVPT,NFPT,VWT,VFWT,VPLOC,FPLOC)                           TP2 Ø161
      WRITE(6,8) S                                                    TP2 Ø162
    8 FORMAT(8X,29HWEIGHTED SUM OF DISTANCES IS:,3X,E11.3)            TP2 Ø163
      DO 11 I=1,NCHECK                                                TP2 Ø164
        DO 9 J=1,NVPT                                                 TP2 Ø165
          IX=IY                                                       TP2 Ø166
          CALL RANDU(IX,IY,YFL)                                       TP2 Ø167
          VPLOC(J)=YFL*T                                              TP2 Ø168
    9   CONTINUE                                                      TP2 Ø169
        WRITE(6,1Ø) I                                                 TP2 Ø17Ø
   1Ø   FORMAT(15HØ      CHECK #(,I3,2H):,5X,11HVARIABLE-PO,          TP2 Ø171
     1  18HINT LOCATIONS ARE:)                                        TP2 Ø172
        WRITE(6,5) (VPLOC(J),J=1,NVPT)                                TP2 Ø173
        S=SUM(NVPT,NFPT,VWT,VFWT,VPLOC,FPLOC)                         TP2 Ø174
        WRITE(6,8) S                                                  TP2 Ø175
   11 CONTINUE                                                        TP2 Ø176
      RETURN                                                          TP2 Ø177
      END                                                             TP2 Ø178


      FUNCTION SUM(NVPT,NFPT,VWT,VFWT,VPLOC,FPLOC)                    TP2 Ø179
C                                                                     TP2 Ø18Ø
C THIS SUBROUTINE CALCULATES THE WEIGHTED SUM OF DISTANCES            TP2 Ø181
```

```
C BETWEEN ALL THE VARIABLE POINTS AND FIXED POINTS.                    TP2 Ø182
C THE MEANINGS OF THE INPUT PARAMETERS ARE EXPLAINED IN THE            TP2 Ø183
C SUBROUTINE 'CHECK' AND TEST PROGRAM #2.                              TP2 Ø184
C                                                                      TP2 Ø185
      INTEGER VWT(3Ø,1),VFWT(3Ø,1)                                     TP2 Ø186
      DIMENSION VPLOC(1),FPLOC(1)                                      TP2 Ø187
      SUM=Ø.                                                           TP2 Ø188
      DO 3 I=1,NVPT                                                    TP2 Ø189
        DO 1 J=1,I                                                     TP2 Ø19Ø
          IF (I.NE.J) SUM=SUM+FLOAT(VWT(I,J))                          TP2 Ø191
     1    *ABS(VPLOC(I)-VPLOC(J))                                      TP2 Ø192
    1   CONTINUE                                                       TP2 Ø193
        DO 2 J=1,NFPT                                                  TP2 Ø194
          SUM=SUM+FLOAT(VFWT(I,J))*ABS(VPLOC(I)-FPLOC(J))              TP2 Ø195
    2   CONTINUE                                                       TP2 Ø196
    3 CONTINUE                                                         TP2 Ø197
      RETURN                                                           TP2 Ø198
      END                                                              TP2 Ø199


      SUBROUTINE LOCATE(NVPT,NFPT,VWT,VFWT,POST)                       LOC ØØØØ
C                                                                      LOC ØØØ1
C THIS SUBROUTINE SOLVES THE FOLLOWING ONE-DIMENSIONAL                 LOC ØØØ2
C MULTIFACILITY LOCATION PROBLEM USING AN ALGORITHM DESIGNED           LOC ØØØ3
C BY PICARD, RATLIFF (SEE OPERATIONS RESEARCH 26 (1978),               LOC ØØØ4
C PP.422-433) AND CHEUNG.                                              LOC ØØØ5
C   " GIVEN NFPT FIXED POINTS ON A LINE, DETERMINE THE                 LOC ØØØ6
C     LOCATIONS OF NVPT VARIABLE POINTS SUCH THAT A WEIGHTED           LOC ØØØ7
C     SUM OF THE DISTANCES BETWEEN THE POINTS IS MINIMUM."             LOC ØØØ8
C OUR ALGORITHM TRANSFORMS THE LOCATION PROBLEM TO A SEQUENCE          LOC ØØØ9
C OF MINIMUM-CUT PROBLEMS. A FORTRAN CODE (THE SUBROUTINE              LOC ØØ1Ø
C NETFLO IN "COMBINATORIAL ALGORITHMS", WRITTEN BY NIJENHUIS           LOC ØØ11
C AND WILF) OF DINIC'S ALGORITHM IS USED TO FIND THE MINIMUM           LOC ØØ12
C CUTS.                                                                LOC ØØ13
C TO APPLY THE SUBROUTINE  'LOCATE', THE FIXED POINTS SHOULD           LOC ØØ14
C BE SUBSCRIPTED ON THE LINE IN INCREASING ORDER OF THEIR              LOC ØØ15
C LOCATIONS. THEIR EXACT LOCATIONS ARE NOT REQUIRED.                   LOC ØØ16
C                                                                      LOC ØØ17
      INTEGER COL,SOURCE,SUM                                           LOC ØØ18
      INTEGER POST(1),CVPT(32),CUT(32),ENDPT(4,93Ø)                    LOC ØØ19
      INTEGER VWT(3Ø,1),VFWT(3Ø,1)                                     LOC ØØ2Ø
C                                                                      LOC ØØ21
C ON INPUT                                                             LOC ØØ22
C     NVPT   NUMBER OF VARIABLE POINTS.                                LOC ØØ23
C     NFPT   NUMBER OF FIXED POINTS.                                   LOC ØØ24
C     VWT    A SQUARE MATRIX WHOSE UPPER TRIANGULAR PART               LOC ØØ25
C            CONTAINS THE WEIGHTS BETWEEN THE VARIABLE POINTS.         LOC ØØ26
C            THE WEIGHTS MUST BE NON-NEGATIVE INTEGERS.                LOC ØØ27
C     VFWT   WEIGHTS BETWEEN THE VARIABLE AND FIXED POINTS.            LOC ØØ28
C            THE WEIGHTS MUST BE NON-NEGATIVE INTEGERS.                LOC ØØ29
C ON OUTPUT                                                            LOC ØØ3Ø
C     POST   A VECTOR INDICATING THE OPTIMAL LOCATIONS OF THE          LOC ØØ31
C            VARIABLE POINTS. VARIABLE POINT I WILL COINCIDE           LOC ØØ32
C            WITH THE FIXED POINT WHOSE POSITION IS POST(I).           LOC ØØ33
C FOR WORKING STORAGE                                                  LOC ØØ34
C     (IN THE FOLLOWING, MNVP MEANS 'THE MAXIMUM NUMBER OF             LOC ØØ35
C      VARIABLE POINTS'.)                                              LOC ØØ36
C     CVPT   A VECTOR OF LENGTH MNVP+2. IT REPRESENTS THE              LOC ØØ37
C            VARIABLE POINTS WHICH PARTICIPATE IN THE CURRENT          LOC ØØ38
C            ITERATION. THE VARIABLE POINT CVPT(I) IS THE              LOC ØØ39
C            NODE I OF THE CURRENT NETWORK.                            LOC ØØ4Ø
C     ENDPT  A MATRIX OF DIMENSION 4 X E, WHERE E=MNVP*                LOC ØØ41
C            *(MNVP+1). EACH ARC CORRESPONDS TO A COLUMN OF            LOC ØØ42
C            ENDPT, WHOSE 4 ROWS DENOTE THE INITIAL VERTEX,            LOC ØØ43
C            THE TERMINAL VERTEX, THE CAPACITY AND MAXIMUM             LOC ØØ44
C            FLOW OF THAT ARC RESPECTIVELY.                            LOC ØØ45
C     CUT    A VECTOR OF LENGTH MNVP+2. FOR A MIN-CUT,                 LOC ØØ46
C            CUT(I)=1 MEANS NODE I IS ASSOCIATED WITH THE              LOC ØØ47
C            SOURCE, AND CUT(I)=Ø MEANS NODE I IS ASSOCIATED           LOC ØØ48
C            WITH THE SINK.                                            LOC ØØ49
C                                                                      LOC ØØ5Ø
       IF (NVPT.EQ.1) GO TO 13                                         LOC ØØ51
C                                                                      LOC ØØ52
C INITIALIZE ENDPT FOR THE FIRST ITERATION.                            LOC ØØ53
      SOURCE=NVPT+1                                                    LOC ØØ54
      COL=Ø                                                            LOC ØØ55
```

```
      DO 6 I=1,NVPT                                                    LOC ØØ56
        CVPT(I)=I                                                      LOC ØØ57
        POST(I)=Ø                                                      LOC ØØ58
        DO 5 J=1,NVPT                                                  LOC ØØ59
          COL=COL+1                                                    LOC ØØ6Ø
          ENDPT(1,COL)=I                                               LOC ØØ61
          ENDPT(2,COL)=J                                               LOC ØØ62
          ENDPT(4,COL)=Ø                                               LOC ØØ63
          IF (I-J) 1,2,4                                               LOC ØØ64
    1     ENDPT(3,COL)=VWT(I,J)                                        LOC ØØ65
          GO TO 5                                                      LOC ØØ66
    2     ENDPT(1,COL)=SOURCE                                          LOC ØØ67
          SUM=Ø                                                        LOC ØØ68
          DO 3 K=1,NFPT                                                LOC ØØ69
             SUM=SUM+VFWT(I,K)                                         LOC ØØ7Ø
    3     CONTINUE                                                     LOC ØØ71
          ENDPT(3,COL)=SUM                                             LOC ØØ72
          GO TO 5                                                      LOC ØØ73
    4     ENDPT(3,COL)=VWT(J,I)                                        LOC ØØ74
    5   CONTINUE                                                       LOC ØØ75
    6 CONTINUE                                                         LOC ØØ76
      NN=NVPT                                                          LOC ØØ77
C                                                                      LOC ØØ78
C ITERATIONS                                                           LOC ØØ79
      DO 12 ITER=1,NFPT                                                LOC ØØ8Ø
        NNODE=NN+2                                                     LOC ØØ81
        DO 7 I=1,NN                                                    LOC ØØ82
          COL=COL+1                                                    LOC ØØ83
          ENDPT(1,COL)=I                                               LOC ØØ84
          ENDPT(2,COL)=NNODE                                           LOC ØØ85
          L=CVPT(I)                                                    LOC ØØ86
          ENDPT(3,COL)=2*VFWT(L,ITER)                                  LOC ØØ87
          ENDPT(4,COL)=Ø                                               LOC ØØ88
    7   CONTINUE                                                       LOC ØØ89
C                                                                      LOC ØØ9Ø
C SOLVE A MIN-CUT PROBLEM BY DINIC'S ALGORITHM.                        LOC ØØ91
        CALL NETFLO(NNODE,COL,ENDPT,SOURCE,NNODE,CUT)                  LOC ØØ92
C                                                                      LOC ØØ93
C UPDATE  POST AND INITIALIZE CVPT FOR THE NEXT ITERATION.             LOC ØØ94
        N2=NN                                                          LOC ØØ95
        NN=Ø                                                           LOC ØØ96
        DO 9 I=1,N2                                                    LOC ØØ97
          IF (CUT(I).EQ.1) GO TO 8                                     LOC ØØ98
          L=CVPT(I)                                                    LOC ØØ99
          POST(L)=ITER                                                 LOC Ø1ØØ
          GO TO 9                                                      LOC Ø1Ø1
    8     NN=NN+1                                                      LOC Ø1Ø2
          CVPT(NN)=CVPT(I)                                             LOC Ø1Ø3
    9   CONTINUE                                                       LOC Ø1Ø4
C                                                                      LOC Ø1Ø5
C NN IS THE NUMBER OF VARIABLE POINTS IN THE NEXT ITERATION.           LOC Ø1Ø6
        IF (NN.EQ.Ø) RETURN                                            LOC Ø1Ø7
C                                                                      LOC Ø1Ø8
C INITIALIZE ENDPT FOR THE NEXT ITERATION.                             LOC Ø1Ø9
        SOURCE=NN+1                                                    LOC Ø11Ø
        COL=Ø                                                          LOC Ø111
        IL=Ø                                                           LOC Ø112
        DO 11 I=1,N2                                                   LOC Ø113
          IF(CUT(I).EQ.Ø) GO TO 11                                     LOC Ø114
          IL=IL+1                                                      LOC Ø115
          JL=Ø                                                         LOC Ø116
          DO 1Ø J=1,N2                                                 LOC Ø117
            IF(CUT(J).EQ.Ø) GO TO 1Ø                                   LOC Ø118
            COL=COL+1                                                  LOC Ø119
            JL=JL+1                                                    LOC Ø12Ø
            ENDPT(1,COL)=IL                                            LOC Ø121
            IF(I.EQ.J) ENDPT(1,COL)=SOURCE                             LOC Ø122
            ENDPT(2,COL)=JL                                            LOC Ø123
            L=(I-1)*N2+J                                               LOC Ø124
            ENDPT(3,COL)=ENDPT(3,L)-ENDPT(4,L)                         LOC Ø125
            ENDPT(4,COL)=Ø                                             LOC Ø126
   1Ø     CONTINUE                                                     LOC Ø127
   11   CONTINUE                                                       LOC Ø128
   12 CONTINUE                                                         LOC Ø129
      RETURN                                                           LOC Ø13Ø
C                                                                      LOC Ø131
```

```
C IN CASE THE NUMBER OF VARIABLE POINTS IS 1, OUR PROBLEM IS          LOC Ø132
C THE SAME AS THE 1-MEDIAN LOCATION PROBLEM. OUR ALGORITHM            LOC Ø133
C IS ALSO EQUIVALENT TO THE FOLLOWING WELL-KNOWN SIMPLE               LOC Ø134
C PROCESS FOR SOLVING 1-MEDIAN LOCATION PROBLEMS.                     LOC Ø135
   13 SUM=-VFWT(1,1)                                                  LOC Ø136
      DO 14 I=2,NFPT                                                  LOC Ø137
        SUM=SUM+VFWT(1,I)                                             LOC Ø138
   14 CONTINUE                                                        LOC Ø139
      I=1                                                             LOC Ø14Ø
   15 IF (SUM.LE.Ø) GO TO 16                                          LOC Ø141
      I=I+1                                                           LOC Ø142
      SUM=SUM-VFWT(1,I)*2                                             LOC Ø143
      GO TO 15                                                        LOC Ø144
   16 POST(1)=I                                                       LOC Ø145
      RETURN                                                          LOC Ø146
      END                                                             LOC Ø147


      SUBROUTINE NETFLO(N,E,ENDPT,SOURCE,SINK,CUT)                    NET ØØØØ
C                                                                     NET ØØØ1
C THIS SUBROUTINE SOLVES A MAX-FLOW-MIN-CUT PROBLEM USING             NET ØØØ2
C DINIC'S ALGORITHM. THE CODE IS TAKEN FROM "COMBINATORIAL            NET ØØØ3
C ALGORITHMS", ACADEMIC PRESS, WRITTEN BY A. NIJENHUIS                NET ØØØ4
C AND H.S. WILF. THERE ARE THREE MINOR CHANGES: (1) STATEMENT         NET ØØØ5
C LABELS ARE RENUMBERED. (2) SOME DATA DECLARATION STATEMENTS         NET ØØØ6
C AND THE FORMAL PARAMETER LIST HAVE BEEN MODIFIED. (3) THE           NET ØØØ7
C WAY OF SUBSCRIPT CONSTRUCTION FOR SOME OF THE ARRAYS IS             NET ØØØ8
C MODIFIED. THE MODIFICATIONS ARE MAINLY FOR COMPLIANCE WITH          NET ØØØ9
C ANSI FORTRAN STANDARD.                                              NET ØØ1Ø
C THANKS ARE DUE TO THE AUTHORS AND PUBLISHER OF                      NET ØØ11
C 'COMBINATORIAL ALGORITHMS' FOR THEIR PERMISSION TO INCLUDE          NET ØØ12
C THIS SUBROUTINE.                                                    NET ØØ13
C                                                                     NET ØØ14
      INTEGER C,E,P,Q,RD,WR,DELTA,SINK,SOURCE,FLOVAL                  NET ØØ15
      INTEGER ENDPT(4,1),CUT(1)                                       NET ØØ16
      INTEGER CAP(32,32),VERT(32,32),AUX(32)                          NET ØØ17
C                                                                     NET ØØ18
C ENDPT AND CUT ARE EXPLAINED IN THE SUBROUTINE 'LOCATE'.             NET ØØ19
C CAP, VERT AND AUX ARE WORKING STORAGE, WHOSE LENGTH IS              NET ØØ2Ø
C EQUAL TO THE NUMBER OF VARIABLE POINTS PLUS 2.                      NET ØØ21
C                                                                     NET ØØ22
C INITIALIZATION.                                                     NET ØØ23
      DO 2 I=1,N                                                      NET ØØ24
        DO 1 J=1,N                                                    NET ØØ25
          CAP(I,J)=Ø                                                  NET ØØ26
    1   CONTINUE                                                      NET ØØ27
    2 CONTINUE                                                        NET ØØ28
      DO 3 I=1,E                                                      NET ØØ29
        I1=ENDPT(1,I)                                                 NET ØØ3Ø
        I2=ENDPT(2,I)                                                 NET ØØ31
        CAP(I1,I2)=ENDPT(3,I)                                         NET ØØ32
    3 CONTINUE                                                        NET ØØ33
      DO 5 I=1,N                                                      NET ØØ34
        K=Ø                                                           NET ØØ35
        DO 4 J=1,N                                                    NET ØØ36
          IF(CAP(I,J)+CAP(J,I).EQ.Ø) GO TO 4                          NET ØØ37
          K=K+1                                                       NET ØØ38
          VERT(I,K)=J                                                 NET ØØ39
    4   CONTINUE                                                      NET ØØ4Ø
        VERT(I,N)=K                                                   NET ØØ41
    5 CONTINUE                                                        NET ØØ42
      NMIN=-N-1                                                       NET ØØ43
      FLOVAL=Ø                                                        NET ØØ44
C                                                                     NET ØØ45
C SCANNING AND LABELING                                               NET ØØ46
    6 LBLSNK=N                                                        NET ØØ47
      DO 7 I=1,N                                                      NET ØØ48
        CUT(I)=NMIN                                                   NET ØØ49
    7 CONTINUE                                                        NET ØØ5Ø
      RD=Ø                                                            NET ØØ51
      WR=Ø                                                            NET ØØ52
      P=SOURCE                                                        NET ØØ53
      LABEL=-1                                                        NET ØØ54
    8 M=VERT(P,N)                                                     NET ØØ55
      I=1                                                             NET ØØ56
    9 IF(I.GT.M) GO TO 13                                             NET ØØ57
```

```
      Q=VERT(P,I)                                                    NET ØØ58
      IF(CAP(P,Q).EQ.Ø) GO TO 12                                     NET ØØ59
      IF(Q.EQ.SINK) LBLSNK=-LABEL                                    NET ØØ6Ø
      IF(CUT(Q)-LABEL) 1Ø,11,12                                      NET ØØ61
   1Ø CUT(Q)=LABEL                                                   NET ØØ62
      WR=WR+1                                                        NET ØØ63
      AUX(WR)=Q                                                      NET ØØ64
   11 I=I+1                                                          NET ØØ65
      GO TO 9                                                        NET ØØ66
   12 VERT(P,I)=VERT(P,M)                                            NET ØØ67
      VERT(P,M)=Q                                                    NET ØØ68
      M=M-1                                                          NET ØØ69
      GO TO 9                                                        NET ØØ7Ø
   13 CUT(P)=M                                                       NET ØØ71
      RD=RD+1                                                        NET ØØ72
      IF(RD.GT.WR) GO TO 24                                          NET ØØ73
      P=AUX(RD)                                                      NET ØØ74
      IF(CUT(P)+LBLSNK.EQ.Ø) GO TO 14                                NET ØØ75
      LABEL=CUT(P)-1                                                 NET ØØ76
      GO TO 8                                                        NET ØØ77
C                                                                    NET ØØ78
C CONSTRUCTION OF PATH FROM SOURCE TO SINK                           NET ØØ79
   14 Q=SOURCE                                                       NET ØØ8Ø
      K=Ø                                                            NET ØØ81
   15 K=K+1                                                          NET ØØ82
      AUX(K)=Q                                                       NET ØØ83
      IF(K.GT.LBLSNK) GO TO 19                                       NET ØØ84
   16 P=AUX(K)                                                       NET ØØ85
   17 M=CUT(P)                                                       NET ØØ86
      IF(M.EQ.Ø) GO TO 18                                            NET ØØ87
      Q=VERT(P,M)                                                    NET ØØ88
      GO TO 15                                                       NET ØØ89
   18 K=K-1                                                          NET ØØ9Ø
      IF(K.EQ.Ø) GO TO 6                                             NET ØØ91
      P=AUX(K)                                                       NET ØØ92
      CUT(P)=CUT(P)-1                                                NET ØØ93
      GO TO 17                                                       NET ØØ94
   19 IF(Q.NE.SINK) GO TO 18                                         NET ØØ95
      DELTA=CAP(P,Q)                                                 NET ØØ96
      DO 2Ø I=2,K                                                    NET ØØ97
        I1=AUX(I-1)                                                  NET ØØ98
        I2=AUX(I)                                                    NET ØØ99
        DELTA=MINØ(DELTA,CAP(I1,I2))                                 NET Ø1ØØ
   2Ø CONTINUE                                                       NET Ø1Ø1
   21 K=K-1                                                          NET Ø1Ø2
      IF(K.EQ.Ø) GO TO 23                                            NET Ø1Ø3
      P=AUX(K)                                                       NET Ø1Ø4
      C=CAP(P,Q)-DELTA                                               NET Ø1Ø5
      IF(C.GT.Ø) GO TO 22                                            NET Ø1Ø6
      CUT(P)=CUT(P)-1                                                NET Ø1Ø7
      KO=K                                                           NET Ø1Ø8
   22 CAP(P,Q)=C                                                     NET Ø1Ø9
      CAP(Q,P)=CAP(Q,P)+DELTA                                        NET Ø11Ø
      Q=P                                                            NET Ø111
      GO TO 21                                                       NET Ø112
   23 FLOVAL=FLOVAL+DELTA                                            NET Ø113
      K=KO                                                           NET Ø114
      GO TO 16                                                       NET Ø115
C                                                                    NET Ø116
C EXIT PROCEDURE.                                                    NET Ø117
   24 DO 25 I=1,N                                                    NET Ø118
        CUT(I)=MINØ(1,CUT(I)-NMIN)                                   NET Ø119
   25 CONTINUE                                                       NET Ø12Ø
      DO 26 I=1,E                                                    NET Ø121
        I1=ENDPT(1,I)                                                NET Ø122
        I2=ENDPT(2,I)                                                NET Ø123
        ENDPT(4,I)=ENDPT(3,I)-CAP(I1,I2)                             NET Ø124
   26 CONTINUE                                                       NET Ø125
      RETURN                                                         NET Ø126
      END                                                            NET Ø127
```

```
 18
  3  4
  Ø  3  2
  3  Ø  1
```

```
 2  1  Ø
 3  2  3  1
 1  2  1  2
 9  1  2  2
 3  4
 Ø  1  3
 1  Ø  3
 3  3  Ø
 9  1  1  2
 2  1  3  2
 2  2  4  1
 5 2Ø
 Ø 21 14  9  7
21  Ø 34 24 32
14 34  Ø 14 36
 9 24 14  Ø  8
 7 32 36  8  Ø
5Ø 49 81 46 52 8Ø 83 48  8 33 99  Ø 5Ø 76 38 4Ø 7Ø 32 3Ø 92
71 63 77 31 62  5  2 85 13 37 19 64 17 59 49 75 52 44 88 94
73 32 27 72  2  7 66 79  1 19 22 76 5Ø 68 56 87 83 69 58 36
97 24 64 49 52  1 74  8 88 84 53 84 46 74 92 45 29 19 54 71
26 51 46 49 93 34 46 35 19 28 21 15 24 31 41 89 22 92 97 99
 5 2Ø
 Ø 21 14  9  7
21  Ø 34 24 32
14 34  Ø 14 36
 9 24 14  Ø  8
 7 32 36  8  Ø
33  Ø 5Ø 48 46 38 32 92 8Ø 99 3Ø  8 4Ø 7Ø 52 5Ø 49 83 76 81
37 64 71 85 31 49 44 94  5 19 88 13 75 52 62 17 63  2 59 77
19 76 73 79 72 56 69 36  7 22 58  1 87 83  2 5Ø 32 66 68 27
84 84 97  8 49 92 19 71  1 53 54 88 45 29 52 46 24 74 74 64
28 15 26 35 49 41 92 99 34 21 97 19 89 22 93 24 51 46 31 46
 5  3
 Ø  2  2  2  2
 2  Ø 2Ø  1  Ø
 2 2Ø  Ø  Ø  Ø
 2  1  Ø  Ø 4Ø
 2  Ø  Ø 4Ø  Ø
1Ø  Ø  Ø
 4  1  4
 4  1  4
 4  5  4
 4  5  4
 3  5
 Ø  1  1
 1  Ø  1
 1  1  Ø
 1  1  6  1  6
 4  1  1  1  1
 1  1  1  1  1
 3  5
 Ø  1  Ø
 1  Ø  1
 Ø  1  Ø
 3  2  2  Ø  Ø
 2  2  2  4  4
 4  Ø  2  6  4
 3  5
 Ø  1  Ø
 1  Ø  1
 Ø  1  Ø
 2  3  Ø  2  Ø
 2  2  4  2  4
 2  4  6  Ø  4
 1 15
 1
 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
 1 15
 1
 1  2  3  4  5  6  7  8  9 1Ø 11 12 13 14 15
 1 2Ø
 1
 4  2 15  9  Ø 45  3  7  5  Ø  Ø  1 5Ø  4  Ø 12 32  4 22  7
 5 15
 Ø  1  1  1  1
```

```
  1  Ø  1  1  1
  1  1  Ø  1  1
  1  1  1  Ø  1
  1  1  1  1  Ø
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  5 15
  Ø  1  1  1  1
  1  Ø  1  1  1
  1  1  Ø  1  1
  1  1  1  Ø  1
  1  1  1  1  Ø
  1  2  3  4  5  6  7  8  9 1Ø 11 12 13 14 15
  1  2  3  4  5  6  7  8  9 1Ø 11 12 13 14 15
  1  2  3  4  5  6  7  8  9 1Ø 11 12 13 14 15
  1  2  3  4  5  6  7  8  9 1Ø 11 12 13 14 15
  1  2  3  4  5  6  7  8  9 1Ø 11 12 13 14 15
  6 2Ø
  Ø  5  2  3 12  8
  5  Ø  6 4Ø  8  6
  2  6  Ø  1 18  7
  3 4Ø  1  Ø  9 13
 12  8 18  9  Ø  4
  8  6  7 13  4  Ø
  4  2 15  9  Ø 45  3  7  5  Ø  Ø  1 5Ø  4  Ø 12 32  4 22  7
  4  2 15  9  Ø 45  3  7  5  Ø  Ø  1 5Ø  4  Ø 12 32  4 22  7
  4  2 15  9  Ø 45  3  7  5  Ø  Ø  1 5Ø  4  Ø 12 32  4 22  7
  4  2 15  9  Ø 45  3  7  5  Ø  Ø  1 5Ø  4  Ø 12 32  4 22  7
  4  2 15  9  Ø 45  3  7  5  Ø  Ø  1 5Ø  4  Ø 12 32  4 22  7
  4  2 15  9  Ø 45  3  7  5  Ø  Ø  1 5Ø  4  Ø 12 32  4 22  7
  6 2Ø
  Ø  5  2  3 12  8
  5  Ø  6 4Ø  8  6
  2  6  Ø  1 18  7
  3 4Ø  1  Ø  9 13
 12  8 18  9  Ø  4
  8  6  7 13  4  Ø
  Ø  Ø  Ø  1  2  Ø  4  Ø  Ø  Ø  9  5 12  Ø  2  8  Ø  2 8Ø 99
  Ø  Ø  1  Ø  7  8  2  4  9  2  Ø 1Ø  4  6  8  6  Ø1ØØ 24 73
  Ø  2  1  2  Ø  3  4  5  Ø  1  6  9  1  Ø  5  2  Ø  4  Ø153
  9  8  Ø  Ø  Ø  Ø  Ø  4  3  Ø  5  3  Ø  1  1  1  Ø  6 49 88
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 22
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  6 2Ø
  Ø  5  2  3 12  8
  5  Ø  6 4Ø  8  6
  2  6  Ø  1 18  7
  3 4Ø  1  Ø  9 13
 12  8 18  9  Ø  4
  8  6  7 13  4  Ø
  Ø  Ø  1  Ø  7  8  2  4  9  2 98 1Ø  4  6  8  6  Ø  1  1  2
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  Ø  Ø  Ø  1  2  Ø  4  Ø  Ø111  9  5 12  Ø  2  8  Ø  2  Ø  Ø
  Ø  2  1  2  Ø  3  4  5  Ø 57  6  9  1  Ø  5  2  Ø  4  Ø  8
  9  8  Ø  Ø  Ø  Ø  Ø  4  3  1Ø25  3  Ø  1  1  1  Ø  6  Ø  Ø
 24  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  6 2Ø
  Ø  1  1  1  1  1
  1  Ø  1  1  1  1
  1  1  Ø  1  1  1
  1  1  1  Ø  1  1
  1  1  1  1  Ø  1
  1  1  1  1  1  Ø
147  Ø  Ø  1  2  Ø  4  Ø  Ø  Ø  9  5 12  Ø  2  8  Ø  2  Ø  Ø
28Ø  Ø  1  Ø  7  8  2  4  9  2  Ø 1Ø  4  6  8  6  Ø  2  3  Ø
23Ø 22  Ø  Ø  Ø  Ø  Ø  4  3  Ø  5  3  Ø  1  1  1  Ø  6  1  Ø
 3Ø 2Ø  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  4
  4 99  1  2  Ø  3  4  5  Ø  1  6  9  1  Ø  5  2  Ø  4  Ø  Ø
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  6 2Ø
  Ø  1  1  1  1  1
  1  Ø  1  1  1  1
  1  1  Ø  1  1  1
```

```
  1  1  1  Ø  1  1
  1  1  1  1  Ø  1
  1  1  1  1  1  Ø
28Ø  Ø  1  Ø  7  8  2  4  9  2  Ø 1Ø  4  6  8  6  Ø  2  3  Ø
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  4 99  1  2  Ø  3  4  5  Ø  1  6  9  1  Ø  5  2  Ø  4  Ø  Ø
 3Ø 2Ø  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  4
147  Ø  Ø  1  2  Ø  4  Ø  Ø  Ø  9  5 12  Ø  2  8  Ø  2  Ø  Ø
23Ø 22  Ø  Ø  Ø  Ø  Ø  4  3  Ø  5  3  Ø  1  1  1  Ø  6  1  Ø
  6 2Ø
  Ø  1  1  1  1  1
  1  Ø  1  1  1  1
  1  1  Ø  1  1  1
  1  1  1  Ø  1  1
  1  1  1  1  Ø  1
  1  1  1  1  1  Ø
  4  2 15  9  Ø 45  3  7  5  Ø  Ø  1 5Ø  4  Ø 12 32  4 22  7
  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  1  1  1  1
 3Ø 2Ø  1  1  1  1 22 15  Ø  3  1  1  1  1  1  1 21  1  1  4
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  Ø  Ø  Ø  1  2  Ø  4  Ø  Ø  Ø  9  5 12  Ø  2  8  Ø  2  Ø 24
  4 99  1  2  Ø  3  4  5  Ø  1  6  9  1  Ø  5  2  Ø  4  Ø  Ø


  7
  2  5 1Ø 5Ø5ØØ1ØØ.ØØ
  3  5 1Ø 3Ø4ØØ2ØØ.ØØ
  3  4 1Ø 5Ø5ØØ15Ø.ØØ
 1Ø 2Ø 2Ø 5Ø5ØØ2ØØ.ØØ
 15 2Ø 25 5Ø5ØØ3ØØ.ØØ
 2Ø 3Ø 3Ø 254ØØ25Ø.ØØ
 3Ø 5Ø 5Ø 3Ø5ØØ5ØØ.ØØ
```

ALGORITHM 559
The Stationary Point of a Quadratic Function Subject to Linear Constraints [E4]

J. T. BETTS
The Aerospace Corporation

Key Words and Phrases: quadratic programming, orthogonal decomposition
CR Categories: 5.14, 5.15, 5.4, 5.40, 5.41
Language: Fortran

DESCRIPTION

This algorithm implements the method developed in [1].

REFERENCE

1. BETTS, J.T. A compact algorithm for computing the stationary point of a quadratic function subject to linear constraints. *ACM Trans. Math. Softw. 6*, 3 (Sept. 1980), 391-397.

ALGORITHM

```
C     PROGRAM DRIVER(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      DIMENSION A(4,4), B(4), C(4,4), D(4), G(4), H(4), U(4), IP(4),
     *  X(4)
      DIMENSION DJNORM(1)
      DIMENSION DMCX(4), AA(4,4), BB(4), CC(4,4), DD(4)
      DIMENSION NN(12), MM(12), KEYKEY(12)
      DATA NN     /Ø,1,1,3,3,3,3,3,4,4,4,4/
      DATA MM     /Ø,Ø,1,Ø,1,2,3,4,Ø,3,Ø,2/
      DATA KEYKEY /1,1,1,1,1,1,1,1,2,2,3,3/
      DATA MAXRA, MAXRC, MAXCAS /4,4,12/
      DATA TAU /1.E-12/
C
C         THE PURPOSE OF THIS DRIVER IS TO DEMONSTRATE THE USE OF
C         SUBROUTINE HSQP
C
C         TEST CASES ARE SPECIFIED BY THE CONTENTS OF THE
C         ARRAYS NN(), MM(), AND KEYKEY().
C         KEY = 1 SETS A AND C NONSINGULAR.
C             = 2 SETS RANK OF A = MIN( N, 2)
C                 AND RANK OF C = MIN( M, N, 2)
C             = 3 SETS C NONSINGULAR AND ALL ELTS OF A = Ø.
C
C         SET TAU TO ABOUT 1ØØ TIMES THE MACHINE PRECISION.
C
      DO 23Ø KASE=1,MAXCAS
      N = NN(KASE)
      M = MM(KASE)
      KEY = KEYKEY(KASE)
```

Received 23 September 1977; revised 30 July 1979; accepted 11 December 1979.

Author's address: The Aerospace Corporation, P.O. Box 92957, Los Angeles, CA 90009.

```
      WRITE (6,1ØØ1) KASE, N, M, KEY, TAU
C
C         DEFINE OBJECTIVE FUNCTION
C
      IF (N.LE.Ø) GO TO 16Ø
      DO 6Ø I=1,N
      B(I) = FLOAT(I) - .Ø5 * FLOAT(I)**2
      BB(I) = B(I)
      DO 5Ø J=I,N
      GO TO (1Ø, 2Ø, 3Ø), KEY
 1Ø   A(I,J) = 1./(FLOAT(J-I)+.3+.Ø1*FLOAT(I))
      GO TO 4Ø
 2Ø   A(I,J) = FLOAT(I+J)
      GO TO 4Ø
 3Ø   A(I,J) = Ø.
 4Ø   CONTINUE
      A(J,I) = A(I,J)
      AA(I,J) = A(I,J)
      AA(J,I) = A(J,I)
 5Ø   CONTINUE
 6Ø   CONTINUE
C
C         DEFINE CONSTRAINTS
C
      IF (M.LE.Ø) GO TO 12Ø
      DO 11Ø I=1,M
      D(I) = FLOAT(I+1) - .Ø5 * FLOAT(I+1)
      DD(I) = D(I)
      DO 1ØØ J=1,N
      C(I,J) = FLOAT(I-J+4)
      GO TO (7Ø, 8Ø, 7Ø), KEY
 7Ø   C(I,J) = 1./(FLOAT(J-I)+.2+.Ø1*FLOAT(I+J))
      GO TO 9Ø
 8Ø   C(I,J) = FLOAT(I-J+4)
 9Ø   CONTINUE
      CC(I,J) = C(I,J)
 1ØØ  CONTINUE
 11Ø  CONTINUE
 12Ø  CONTINUE
      WRITE (6,1ØØ2)
      DO 13Ø I=1,N
      WRITE (6,1ØØ8) (A(I,J),J=1,N), B(I)
 13Ø  CONTINUE
      IF (M.LE.Ø) GO TO 15Ø
      WRITE (6,1ØØ3)
      DO 14Ø I=1,M
      WRITE (6,1ØØ8) (C(I,J),J=1,N), D(I)
 14Ø  CONTINUE
 15Ø  CONTINUE
 16Ø  CONTINUE
C
C         CALL HSQP TO SOLVE PROBLEM
C
      CALL HSQP(A, B, C, D, M, N, TAU, G, H, U, IP, MAXRA, MAXRC,
     *  DJNORM, X, KRANK)
C
C         WRITE SOLUTION VECTOR
C
      WRITE (6,1ØØ7) DJNORM(1), KRANK
      IF (DJNORM(1).LT.Ø. .OR. N.LE.Ø) GO TO 22Ø
      WRITE (6,1ØØ6) (X(I),I=1,N)
C
C         CHECK CONSTRAINTS.
C
      IF (M.LE.Ø) GO TO 19Ø
      DO 18Ø I=1,M
      DMCX(I) = DD(I)
      DO 17Ø J=1,N
      DMCX(I) = DMCX(I) - CC(I,J)*X(J)
 17Ø  CONTINUE
 18Ø  CONTINUE
      WRITE (6,1ØØ4) (DMCX(I),I=1,M)
 19Ø  CONTINUE
C
C         EVALUATE OBJECTIVE FUNCTION.
C
```

```
      VALUE = Ø.
      DO 21Ø J=1,N
      SUM = Ø.
      DO 2ØØ I=1,N
      SUM = SUM + X(I)*AA(I,J)
 2ØØ  CONTINUE
      VALUE = VALUE + (Ø.5*SUM+BB(J))*X(J)
 21Ø  CONTINUE
      WRITE (6,1ØØ5) VALUE
 22Ø  CONTINUE
 23Ø  CONTINUE
      STOP
 1ØØ1 FORMAT (51HØ**************************************************/
     */31H  CASE    N    M  KEY       TAU/1X,4I5,E1Ø.3)
 1ØØ2 FORMAT (26HØ AUGMENTED MATRIX (A:B) =)
 1ØØ3 FORMAT (26HØ AUGMENTED MATRIX (C:D) =)
 1ØØ4 FORMAT (33HØ CONSTRAINT RESIDUALS  D - C*X =/(1X, 8E14.3))
 1ØØ5 FORMAT (32HØ VALUE OF OBJECTIVE FUNCTION = , E2Ø.6)
 1ØØ6 FORMAT (16HØSOLUTION VECTOR/(1X, 8E14.6))
 1ØØ7 FORMAT (24HØPROJECTED GRADIENT NORM, 5X, E2Ø.8/ 13HØRANK OF PROJ,
     *  2ØHECTED HESSIAN MATRIX, 5X, I1Ø)
 1ØØ8 FORMAT (1X, 8E14.6)
      END


      SUBROUTINE HSQP(A,B,C,D,M,N,TAU,G,H,U,IP,MAXRA,MAXRC,DJNORM,X,
     $    KRANK)
C
C
      DIMENSION B(1),D(1),G(1),H(1),U(1),IP(1),DJNORM(1),X(1)
      DIMENSION A(MAXRA,1),C(MAXRC,1)
C
C         PROGRAMMER AND DATE:  J.T.BETTS, JAN. 1978.
C
C         PURPOSE:  GIVEN AN M X N MATRIX C (OF RANK M), AN M VECTOR D,
C         AN N X N SYMMETRIC MATRIX A, AND AN N VECTOR B, FIND THE
C         STATIONARY POINT X OF THE QUADRATIC
C
C         J = .5*(X**T)*A*X + (B**T)*X
C
C         SUBJECT TO THE CONSTRAINTS
C
C         C*X = D,
C
C         IF A STATIONARY POINT DOES NOT EXIST THE ALGORITHM WILL FIND
C         A POINT WHICH SATISFIES THE CONSTRAINTS AND MINIMIZES THE
C         NORM OF THE GRADIENT OF J PROJECTED ON THE CONSTRAINT SURFACE.
C
C         ALGORITHM;  ORTHOGONAL DECOMPOSITION OF C MATRIX USING
C         HOUSEHOLDER TRANSFORMATIONS, FOLLOWED BY APPLICATION OF THE
C         OPTIMALITY CONDITIONS IN THE REDUCED VARIABLES.
C
C         INPUT:
C
C            A      N X N SYMMETRIC HESSIAN MATRIX
C            B      N DIMENSIONAL GRADIENT VECTOR
C            C      M X N JACOBIAN MATRIX (RANK M)
C            D      M DIMENSIONAL CONSTRAINT VECTOR
C            M      THE NUMBER OF CONSTRAINTS
C            N      THE NUMBER OF VARIABLES
C            TAU    PSEUDORANK TEST PARAMETER.  FOR A MACHINE WITH K
C                   SIGNIFICANT FIGURES AN APPROPRIATE VALUE IS
C                   TAU = 1.E-(K-2).
C            G      AUXILLIARY STORAGE (LENGTH M)
C            H      AUXILLIARY STORAGE (LENGTH N-M)
C            U      AUXILLIARY STORAGE (LENGTH N-M)
C            IP     AUXILLIARY STORAGE (LENGTH N-M)
C            MAXRA  MAXIMUM ROW DIMENSION OF A (MAXRA X N)
C            MAXRC  MAXIMUM ROW DIMENSION OF C (MAXRC X M)
C
C         OUTPUT:
C
C            DJNORM    IF DJNORM .GE. Ø. IT IS THE NORM OF THE PROJECTED
```

```
C                   GRADIENT OF THE CONSTRAINED QUADRATIC FORM.
C                   IF DJNORM = Ø. X IS A STATIONARY POINT OF THE
C                   CONSTRAINED QUADRATIC FORM.
C                   DJNORM = -1. MEANS INPUT ERRORS.      NO X COMPUTED.
C                   DJNORM = -2. MEANS RANK(C) .LT. M.    NO X COMPUTED.
C            X      COMPUTED SOLUTION VECTOR.  IT IS A STATIONARY POINT
C                   IF DJNORM = Ø.
C            KRANK  PSEUDORANK OF PROJECTED HESSIAN MATRIX (K2**T)*A*K2.
C                   IF KRANK = N-M, X IS THE UNIQUE SOLUTION.
C                   IF KRANK .LT. N-M, X IS THE MINIMUM LENGTH SOLUTION.
C
C         NOTE:     THE INPUT VALUES OF A,B,C, AND D ARE DESTROYED.
C
C ------------------------------------------------------------------
C
C         INITIALIZATION
C
      KRANK = Ø
      MP1 = M + 1
      NMM = N - M
      DJNORM(1) = -1.
C
C         CHECK FOR INPUT ERRORS
C
      IF(N.LE.Ø.OR.N.GT.MAXRA.OR.M.GT.MAXRC.OR.M.GT.N
     $  .OR. M .LT. Ø) RETURN
C
C         COMPUTE TOLERANCE, ATOL, FOR PSEUDORANK OF A.
C
      TEMP = Ø.
      DO 2Ø J = 1,N
      SUMSQ = Ø.
      DO 1Ø I = 1,N
      SUMSQ = SUMSQ + A(I,J)**2
 1Ø   CONTINUE
      TEMP = AMAX1(TEMP,SUMSQ)
 2Ø   CONTINUE
      ATOL = TAU*SQRT(TEMP)
C
C         IF THE PROBLEM IS UNCONSTRAINED GO TO STEP 6
C
      IF(M.EQ.Ø) GO TO 14Ø
C
C ------------------------------------------------------------------
C
C         STEP. 1.  COMPUTE ORTHOGONAL MATRIX K.  TRIANGULARIZE C.
C
      DO 3Ø I = 1,M
      CALL H12(1,I,I+1,N,C(I,1),MAXRC,G(I),C(I+1,1),MAXRC,1,M-I)
 3Ø   CONTINUE
C
C ------------------------------------------------------------------
C
C         STEP 2.  COMPUTE Y1HAT BY SOLVING THE LOWER TRIANGULAR
C         SYSTEM C*Y1 = D.  STORE IN X.
C
      TEMP = Ø.
      DO 5Ø J=1,M
      DO 4Ø I=J,M
      TEMP=AMAX1(TEMP,ABS(C(I,J)))
 4Ø   CONTINUE
 5Ø   CONTINUE
      CTOL = TAU*TEMP
      DO 9Ø I = 1,M
      IM1 = I - 1
      X(I) = D(I)
      IF(I .EQ. 1) GO TO 7Ø
      DO 6Ø J = 1,IM1
      X(I) = X(I) - C(I,J)*X(J)
 6Ø   CONTINUE
 7Ø   CONTINUE
```

```
      IF(ABS(C(I,I)) .GT. CTOL) GO TO 8Ø
      DJNORM(1) = -2.
      RETURN
 8Ø   CONTINUE
      X(I) = X(I)/C(I,I)
 9Ø   CONTINUE
C
C         WHEN THERE ARE NO DEGREES OF FREEDOM GO TO STEP 8
C
      IF(M .LT. N) GO TO 1ØØ
      DJNORM(1) = Ø.
      GO TO 19Ø
 1ØØ  CONTINUE
C
C ---------------------------------------------------------------------
C
C         STEP 3.  COMPUTE ATILDA = (K**T)*A
C
      DO 11Ø I = 1,M
      CALL H12(2,I,I+1,N,C(I,1),MAXRC,G(I),A,1,MAXRA,N)
 11Ø  CONTINUE
C
C ---------------------------------------------------------------------
C
C         STEP 4.  FORM THE LAST N-M ROWS OF AHAT = ATILDA*K;  I.E.
C         COMPUTE A21HAT = (K2**T)*A*K1 AND A22HAT = (K2**T)*A*K2
C
      DO 12Ø I = 1,M
      CALL H12(2,I,I+1,N,C(I,1),MAXRC,G(I),A(MP1,1),MAXRA,1,NMM)
 12Ø  CONTINUE
C
C ---------------------------------------------------------------------
C
C         STEP 5.  COMPUTE BTILDA = (K**T)*B
C
      DO 13Ø I = 1,M
      CALL H12(2,I,I+1,N,C(I,1),MAXRC,G(I),B,1,1,1)
 13Ø  CONTINUE
C
C ---------------------------------------------------------------------
C
C         STEP 6.  COMPUTE B2HAT = -B2TILDA - A21HAT*Y1HAT
C
 14Ø  CONTINUE
      DO 17Ø I = MP1,N
      B(I) = -B(I)
      IF(M .EQ. Ø) GO TO 16Ø
      DO 15Ø J = 1,M
      B(I) = B(I) - A(I,J)*X(J)
 15Ø  CONTINUE
 16Ø  CONTINUE
 17Ø  CONTINUE
C
C ---------------------------------------------------------------------
C
C         STEP 7. SOLVE A22HAT*Y2 = B2HAT FOR Y2 USING HFTI
C
C
C
      CALL HFTI(A(MP1,MP1),MAXRA,NMM,NMM,B(MP1),1,1,ATOL,KRANK,
     $    DJNORM,H,U,IP)
C
      DO 18Ø I = MP1,N
      X(I) = B(I)
 18Ø  CONTINUE
C
C         IF THE PROBLEM IS UNCONSTRAINED, RETURN.
C
      IF(M.EQ.Ø) RETURN
C
C ---------------------------------------------------------------------
```

```
C
C          STEP 8.  COMPUTE X = K*Y
C
 19Ø  CONTINUE
      DO 2ØØ K = 1,M
      I = MP1 - K
      CALL H12(2,I,I+1,N,C(I,1),MAXRC,G(I),X,1,1,1)
 2ØØ  CONTINUE
C
      RETURN
      END


      SUBROUTINE H12(MODE,LPIVOT,L1,M,U,IUE,UP,C,ICE,ICV,NCV)
C
C
C         ALGORITHM H12:  C.L. LAWSON AND R.J. HANSON, "SOLVING LEAST
C         SQUARES PROBLEMS", PRENTICE-HALL,1974. APPENDIX C,P. 3Ø8.
C
C
C         PURPOSE:  CONSTRUCTION AND/OR APPLICATION OF A SINGLE
C         HOUSEHOLDER TRANSFORMATION ...  Q = I + U*(U**T)/B
C
C         MODE = 1 OR 2 TO SELECT ALGORITHM H1 OR H2.
C         LPIVOT IS THE INDEX OF THE PIVOT ELEMENT
C         L1,M   IF L1.LE.M THE TRANSFORMATION WILL BE CONSTRUCTED TO
C                ZERO ELEMENTS INDEXED FROM L1 THROUGH M.  IF L1.GT.M
C                THE SUBROUTINE DOES AN IDENTITY TRANSFORMATION.
C         U(),IUE,UP    ON ENTRY TO H1 U() CONTAINS THE PIVOT VECTOR.
C                IUE IS THE STORAGE INCREMENT BETWEEN ELEMENTS.  ON EXIT
C                FROM H1 U() AND UP CONTAIN QUANTITIES DEFINING THE
C                VECTOR U OF THE HOUSEHOLDER TRANSFORMATION. ON ENTRY TO
C                H2 U() AND UP SHOULD CONTAIN QUANTITIES PREVIOUSLY
C                COMPUTED BY H1.  THESE WILL NOT BE MODIFIED BY H2.
C         C()    ON ENTRY TO H1 OR H2 C() CONTAINS A MATRIX WHICH WILL
C                BE REGARDED AS A SET OF VECTORS TO WHICH THE
C                HOUSEHOLDER TRANSFORMATION IS TO BE APPLIED.  ON EXIT
C                C() CONTAINS THE SET OF TRANSFORMED VECTORS.
C         ICE    STORAGE INCREMENT BETWEEN ELEMENTS OF VECTORS IN C()
C         ICV    STORAGE INCREMENT BETWEEN VECTORS IN C().
C         NCV    NUMBER OF VECTORS IN C() TO BE TRANSFORMED. IF NCV.LE.Ø
C                NO OPERATIONS WILL BE DONE ON C().
C
      DIMENSION U(IUE,M),C(1)
      DOUBLE PRECISION SM,B
C
      IF(Ø.GE.LPIVOT.OR.LPIVOT.GE.L1.OR.L1.GT.M) RETURN
      CL = ABS(U(1,LPIVOT))
      IF(MODE.EQ.2) GO TO 6Ø
C
C         CONSTRUCT THE TRANSFORMATION.
C
      DO 1Ø J = L1,M
 1Ø   CL = AMAX1(ABS(U(1,J)),CL)
      IF(CL) 13Ø,13Ø,2Ø
 2Ø   CLINV = 1./CL
      SM = (DBLE(U(1,LPIVOT))*CLINV)**2
      DO 3Ø J = L1,M
 3Ø   SM = SM + (DBLE(U(1,J))*CLINV)**2
C
C         CONVERT DBLE. PREC. SM TO SNGL. PREC. SM1
C
      SM1 = SM
      CL = CL*SQRT(SM1)
      IF(U(1,LPIVOT)) 5Ø,5Ø,4Ø
 4Ø   CL = -CL
 5Ø   UP = U(1,LPIVOT) - CL
      U(1,LPIVOT) = CL
      GO TO 7Ø
C
C         APPLY THE TRANSFORMATION I + U*(U**T)/B TO C
C
 6Ø   IF(CL) 13Ø,13Ø,7Ø
 7Ø   IF(NCV.LE.Ø) RETURN
      B = DBLE(UP)*U(1,LPIVOT)
```

```
C
C          B MUST BE NONPOSITIVE HERE.  IF B=Ø RETURN
C
      IF(B) 8Ø,13Ø,13Ø
 8Ø   B = 1.DØ/B
      I2 = 1 - ICV + ICE*(LPIVOT-1)
      INCR = ICE*(L1-LPIVOT)
      DO 12Ø J = 1,NCV
      I2 = I2 + ICV
      I3 = I2 + INCR
      I4 = I3
      SM = C(I2)*DBLE(UP)
      DO 9Ø I = L1,M
      SM = SM + C(I3)*DBLE(U(1,I))
 9Ø   I3 = I3 + ICE
      IF(SM) 1ØØ,12Ø,1ØØ
 1ØØ  SM = SM*B
      C(I2) = C(I2) + SM*DBLE(UP)
      DO 11Ø I = L1,M
      C(I4) = C(I4) + SM*DBLE(U(1,I))
 11Ø  I4 = I4 + ICE
 12Ø  CONTINUE
 13Ø  RETURN
      END


      SUBROUTINE HFTI(A,MDA,M,N,B,MDB,NB,TAU,KRANK,RNORM,H,G,IP)
C
C
C          PURPOSE:  SOLVE THE MATRIX LINEAR LEAST SQUARE PROBLEM
C                              MIN NORM(A*X - B)
C                    WHERE A IS MXN,  B IS MXNB,  X IS NXNB.
C
C      REF.  C.L. LAWSON AND R.J. HANSON, "SOLVING LEAST SQUARES PROBLEMS
C          PRENTICE-HALL,1974.   ALGORITHM HFTI, APPENDIX C.,P.29Ø.
C
C
      DIMENSION A(MDA,N),B(MDB,NB),H(N),G(N),IP(N),RNORM(NB)
      DOUBLE PRECISION SM,DZERO
C
      SZERO = Ø.
      DZERO = Ø.DØ
      FACTOR = .ØØ1
C
      K = Ø
      LDIAG = MINØ(M,N)
      IF(LDIAG.LE.Ø) GO TO 27Ø
C
      DO 8Ø J = 1,LDIAG
      IF(J.EQ.1) GO TO 2Ø
C
C          UPDATE SQUARED COLUMN LENGTHS AND FIND LMAX
C
      LMAX = J
      DO 1Ø L = J,N
      H(L) = H(L) - A(J-1,L)**2
      IF(H(L).GT.H(LMAX)) LMAX = L
 1Ø   CONTINUE
      IF(FACTOR*H(LMAX)) 2Ø,2Ø,5Ø
C
C          COMPUTE SQUARED COLUMN LENGTHS AND FIND LMAX
C
 2Ø   LMAX = J
      DO 4Ø L = J,N
      H(L) = Ø.
      DO 3Ø I = J,M
 3Ø   H(L) = H(L) + A(I,L)**2
      IF(H(L).GT.H(LMAX)) LMAX = L
 4Ø   CONTINUE
      HMAX = H(LMAX)
C
C          LMAX HAS BEEN DETERMINED.  DO COLUMN INTERCHANGES IF NEEDED.
C
 5Ø   CONTINUE
      IP(J) = LMAX
      IF(IP(J).EQ.J) GO TO 7Ø
```

```
      DO 6Ø I = 1,M
      TMP = A(I,J)
      A(I,J) = A(I,LMAX)
 6Ø   A(I,LMAX) = TMP
      H(LMAX) = H(J)
C
C         COMPUTE THE J-TH TRANSFORMATION AND APPLY IT TO A AND B.
C
 7Ø   CALL H12(1,J,J+1,M,A(1,J),1,H(J),A(1,J+1),1,MDA,N-J)
 8Ø   CALL H12(2,J,J+1,M,A(1,J),1,H(J),B,1,MDB,NB)
C
C         DETERMINE THE PSEUDORANK,K, USING THE TOLERANCE, TAU
C
      DO 9Ø J = 1,LDIAG
      IF(ABS(A(J,J)).LE.TAU) GO TO 1ØØ
 9Ø   CONTINUE
      K = LDIAG
      GO TO 11Ø
 1ØØ  K = J - 1
 11Ø  KP1 = K + 1
C
C         COMPUTE THE NORMS OF THE RESIDUAL VECTORS.
C
      IF(NB.LE.Ø) GO TO 14Ø
      DO 13Ø JB = 1,NB
      TMP = SZERO
      IF(KP1.GT.M) GO TO 13Ø
      DO 12Ø I = KP1,M
 12Ø  TMP = TMP + B(I,JB)**2
 13Ø  RNORM(JB) = SQRT(TMP)
 14Ø  CONTINUE
C
C         SPECIAL FOR PSEUDORANK = Ø.
C
      IF(K.GT.Ø) GO TO 16Ø
      IF(NB.LE.Ø) GO TO 27Ø
      DO 15Ø JB = 1,NB
      DO 15Ø I = 1,N
 15Ø  B(I,JB) = SZERO
      GO TO 27Ø
C
C         IF THE PSEUDORANK IS LESS THAN N COMPUTE HOUSEHOLDER
C         DECOMPOSITION OF FIRST K ROWS.
C
 16Ø  IF(K.EQ.N) GO TO 18Ø
      DO 17Ø II = 1,K
      I = KP1 - II
 17Ø  CALL H12(1,I,KP1,N,A(I,1),MDA,G(I),A,MDA,1,I-1)
 18Ø  CONTINUE
C
C
      IF(NB.LE.Ø) GO TO 27Ø
      DO 26Ø JB = 1,NB
C
C         SOLVE THE K BY K TRIANGULAR SYSTEM
C
      DO 21Ø L = 1,K
      SM = DZERO
      I = KP1 - L
      IF(I.EQ.K) GO TO 2ØØ
      IP1 = I + 1
      DO 19Ø J = IP1,K
 19Ø  SM = SM + DBLE(A(I,J))*DBLE(B(J,JB))
 2ØØ  SM1 = SM
 21Ø  B(I,JB) = (B(I,JB)-SM1)/A(I,I)
C
C         COMPLETE COMPUTATION OF SOLUTION VECTOR
C
      IF(K.EQ.N) GO TO 24Ø
      DO 22Ø J = KP1,N
 22Ø  B(J,JB) = SZERO
      DO 23Ø I = 1,K
 23Ø  CALL H12(2,I,KP1,N,A(I,1),MDA,G(I),B(1,JB),1,MDB,1)
C
C         REORDER THE SOLUTION VECTOR TO COMPENSATE FOR THE COLUMN
C         INTERCHANGES.
```

```
      C
       24Ø  DO 25Ø JJ = 1,LDIAG
            J = LDIAG + 1 - JJ
            IF(IP(J).EQ.J) GO TO 25Ø
            L = IP(J)
            TMP = B(L,JB)
            B(L,JB) = B(J,JB)
            B(J,JB) = TMP
       25Ø  CONTINUE
       26Ø  CONTINUE
      C
      C         THE SOLUTION VECTORS, X, ARE NOW IN THE FIRST N ROWS OF THE
      C         ARRAY B(.)
      C
       27Ø  KRANK = K
            RETURN
            END
```

ALGORITHM 560
JNF, An Algorithm for Numerical Computation of the Jordan Normal Form of a Complex Matrix [F2]

BO KÅGSTRÖM and AXEL RUHE
University of Umeå, Sweden

Key Words and Phrases: Jordan normal form, canonical form, eigenvalues, eigenvectors, principal vectors, block diagonal form
CR Categories: 5.14
Language: Fortran

DESCRIPTION

1. Introduction

The routines given here are the actual Fortran implementation of the algorithm presented and discussed in [2]. We describe in detail how to use the Fortran subroutines and how to reach the results from a call. We also give some comments on the code that might be of value when implementing the subroutines on a particular machine. The subroutines have been checked with the PFORT verifier [4] and the notation from [2] is used.

2. The User-Written Routine DECIDE

DECIDE is a user-written subroutine which makes it possible for the user to change the grouping and/or the values of the numerical multiple eigenvalues.

This routine is useful when we have some information on the eigenvalues and their multiplicities in advance, and want the eigenvectors and the principal vectors for the given matrix. For instance, if we know of physical reasons why zero should be the only possible multiple eigenvalue and all others simple, then the contents of the parameters should be changed as indicated in Table I.

If the user does not want to influence the grouping or the values of the eigenvalues, it is enough to define a dummy subroutine, i.e.,

```
SUBROUTINE DECIDE (NM, N, NDEL, CSHTR, CSHTI, NBLOCK, HR, HI)
RETURN
END
```

3. The Rank Determination Process and the RDEFL Routine

The rank determination process is described in [2, Section 2.2 and Section 3, Step 6 of the algorithm]. According to these two alternatives we have two subroutines

Received 21 May 1975 and August 1977.

This work was supported by the Swedish Natural Science Research Council.
Authors' address: Institute of Information Processing, University of Umeå, S-901 87 Umeå, Sweden.

Table I. Example of Use of the Subroutine DECIDE

On entry N = 10 NBLOCK = 6		On exit NBLOCK = 7	
NDEL	CSHT	NDEL	CSHT
0	−1.5	0	−1.5
1	0.002	1	0.0
5	2.4	5	H(6, 6)
7	3.5	6	H(7, 7)
8	8.6	7	3.5
9	9.5	8	8.6
10		9	9.5
		10	

RDEFL—one for the singular value decomposition strategy and one for the Kublanovskaya (RQ) decomposition strategy. Only the routine based on singular value decomposition is presented here.

In the RDEFL routine we use the CSVD algorithm [1] with one change; namely, we sort the singular values in increasing instead of decreasing order as in the original program.

4. How to Reach Any of the Vectors

The eigenvectors and principal vectors are stored in the arrays ZR, ZI(NM, N). To find the vectors, we use the information stored in NBLOCK, NDEL, NDB, NDEFL, and NXT (see the parameter list of subroutine JNF).

The column indices K in ZR and ZI for eigenvectors, principal vectors, and principal chains of an eigenvalue are listed below in Algol for-statement notation.

(a) Find the eigenvectors of the multiple eigenvalue
I = 1, . . . , NBLOCK.
J := NDB[I];
for K := NDEL[I] + 1 **step** 1 **until** NDEFL[J + 1] − 1 **do**;

(b) Find the principal vectors of grade P to the eigenvalue I = 1, . . . , NBLOCK.
J := NDB[I] + P − 1;
for K := NDEFL[J] **step** 1 **until** NDEFL[J + 1] − 1 **do**;

(c) Find the principal chain which ends with the eigenvector in column L.

```
First:
(A − EV[L]*I)*Z[L] = 0   (eigenvector)
Then:
K1 := L;
for K := NXT[K1] while K > 0 do
begin
Here:
(A − EV[K]*I)*Z[K] = SUPD[K1]*Z[K1];   (principal vector)
K1 := K
end;
```

(Here we use the fact that the diagonal elements may be different as in [2, eq. (4.2)]. To get chains of the nilpotent B, EV[K] should be replaced by EV[L].)

In Table II we list the values of NDEL, NDEFL, NXT, NDB, and SUPD for the first test matrix in [2, Section 5].

5. Comments on the Fortran Code

Since we use the EISPACK routines CBAL, COMHES, COMLR2, and CBABK2 [5], we have their representation of complex arrays, i.e., the real parts are represented in one array with the name ending in R (e.g., HR, ZR) and the imaginary parts are represented in one array with the name ending in I (e.g., HI, ZI). This representation makes it possible to execute most of the computations in real arithmetic.

We use real arithmetic in all steps of the algorithm except Steps 4 and 7, i.e., the elimination processes, where we use some complex arithmetic.

Table II. Example of Output of Index Vectors

I	NDEL	NDB	NDEFL	NXT	SUPD
1	0	1	1	3	1.9903
2	4	3	3	4	48.080
3	9	6	5	0	0
4	10	0	7	0	0
5	0	0	9	7	3.1314
6	0	0	10	8	5.5354
7	0	0	11	0	0
8	0	0	0	9	1.3994
9	0	0	0	0	0
10	0	0	0	0	0

Note. Output for the first test matrix in [2] is displayed.

Because of this complex arithmetic we also have to use the complex standard functions REAL(X), AIMAG(X), and CMPLX(X, Y).

If the actual Fortran compiler does not accept complex arithmetic, we can implement the functions CMUL, CDIV, CADD, and CSUB from [6] and substitute the actual statements with new statements.

A new parameter (NOBACK) has been added to the EISPACK routine COMLR2. We use this parameter (NOBACK = 0) as a control so that we do not get the eigenvectors from COMLR2. We get the transformations which transform the matrix to upper triangular form.

In the RDEFL routine we use complex arrays and arithmetic since the routine CSVD1 [1] is made for complex arithmetic.

6. Numerical Experiments

We have performed several numerical tests, both on matrices with a well-defined Jordan normal form, and on cases specially constructed in order to provide the routine with difficulties. Results from full Jordan normal form reduction are reported and discussed in [2]. In [2, Section 4] we describe how to choose tolerance parameters in the grouping procedure and in the procedure for determining the structures inside the invariant subspaces corresponding to different numerical multiple eigenvalues. We also describe how to analyze the results from the program, given a combination of the tolerance parameters.

A large number of results from block diagonal reductions (ISTEP = 5 and 6) are reported and discussed in [3].

REFERENCES

1. BUSINGER, P.A., AND GOLUB, G.H. Singular value decomposition of a complex matrix. Algorithm 358. *Comm. ACM 12* (1969), 564–565.
2. KÅGSTRÖM, B., AND RUHE, A. An algorithm for numerical computation of the Jordan normal form of a complex matrix. *ACM Trans. Math. Softw. 6,* 3 (Sept. 1980), 398–419.
3. KÅGSTRÖM, B. Numerical computation of matrix functions. Rep. UMINF-58.77, Dep. Information Processing, Univ. Umeå, Umeå, Sweden, 1977.
4. RYDER, B.G., AND HALL, A.D. The PFORT verifier. Computing Sci. Tech. Rep. 12, Bell Labs., Murray Hill, N.J.
5. SMITH, B.T., ET AL. *Matrix Eigensystem Routines—EISPACK Guide.* Springer-Verlag, New York, 1974.
6. WILKINSON, J.H., AND REINSCH, C. (Eds.) *Handbook for Automatic Computation II: Linear Algebra.* Springer-Verlag, New York, 1971.

ALGORITHM

[Summary information and a part of the listings are printed here. The complete listing is available from the ACM Algorithms Distribution Service (see inside back cover for order form) or may be found in microfiche form in "Collected Algorithms from ACM."]

NAME(n): indicates a Fortran module with n records

NAME^T(n): indicates "NAME" is included for testing purposes
Contents: MAIN^T(328), RESID^T(75), DECIDE^T(56), MATRIS^T(43), MATRS1^T(32), MATRS3^T(55), MATRS4^T(62), MATRS5^T(42), JNF(834), RDEFL(132), CBAL(209), COMHES(138), COMLR2(405), CBABK2(91), CVSD1(384), OUTPUT^T(33), MATBLK^T(37), MATRS2^T(148)

```
      SUBROUTINE DECIDE (NM,N,NDEL,CSHTR,CSHTI,NBLOCK,HR,HI)          ØØØØ4Ø4Ø
C     ************************************************************    ØØØØ4Ø5Ø
C     *                                                              *  ØØØØ4Ø6Ø
C     *  DECIDE IS A USER WRITTEN SUBROUTINE WHICH MAKES IT POSSIBLE *  ØØØØ4Ø7Ø
C     *  FOR THE USER TO CHANGE THE GROUPING OF THE NUMERICAL MUL-   *  ØØØØ4Ø8Ø
C     *  TIPLE EIGENVALUES,AND/OR CHANGE THE VALUES OF THE MULTIPLE  *  ØØØØ4Ø9Ø
C     *  EIGENVALUES.                                                *  ØØØØ41ØØ
C     *  THE FORMAL PARAMETER LIST                                   *  ØØØØ411Ø
C     *                                                              *  ØØØØ412Ø
C     *  NM        THE ROW DIMENSION OF THE TWO-DIMENSIONAL ARRAY    *  ØØØØ413Ø
C     *            PARAMETERS(HR,HI)AS DECLARED IN THE CALLING       *  ØØØØ414Ø
C     *            PROGRAM DIMENSION STATEMENT                       *  ØØØØ415Ø
C     *  N         THE ORDER OF THE ORIGINAL MATRIX                  *  ØØØØ416Ø
C     *  NBLOCK    THE NUMBER OF BLOCKS I.E THE NUMBER OF NUMERICAL  *  ØØØØ417Ø
C     *            MULTIPLE EIGENVALUES                              *  ØØØØ418Ø
C     *  CSHTR,    THE REAL AND IMAGINARY PARTS,RESPECTIVELY,OF THE  *  ØØØØ419Ø
C     *  CSHTI     NUMERICAL MULTIPLE EIGENVALUES                    *  ØØØØ42ØØ
C     *  NDEL      INDICATES THE MULTIPLICITY OF THE NUMERICAL       *  ØØØØ421Ø
C     *            MULTIPLE EIGENVALUES.NDEL(I+1)-NDEL(I)=THE        *  ØØØØ422Ø
C     *            MULTIPLICITY OF EIGENVALUE I FOR I=1,..,NBLOCK    *  ØØØØ423Ø
C     *  HR,HI     THE REAL AND IMAGINARY PARTS,RESPECTIVELY, OF     *  ØØØØ424Ø
C     *            THE UPPER TRIANGULAR MATRIX RESULTING FROM        *  ØØØØ425Ø
C     *            STEPS 1-3 OF THE ALGORITHM                        *  ØØØØ426Ø
C     *                                                              *  ØØØØ427Ø
C     *  NOTE -- THE USER MUST NOT CHANGE HR,HI                      *  ØØØØ428Ø
C     *                                                              *  ØØØØ429Ø
C     ************************************************************    ØØØØ43ØØ
        INTEGER NDEL(N)                                               ØØØØ431Ø
        REAL HR(NM,N),HI(NM,N),CSHTR(1),CSHTI(1)                      ØØØØ432Ø
        WRITE(6,9)                                                    ØØØØ433Ø
    9   FORMAT(1HØ,1X,47HTHE FOLLOWING OUTPUT (A,B AND C) ARE PRINTED BY,  ØØØØ434Ø
       *          31HTHE USER WRITTEN ROUTINE DECIDE/                 ØØØØ435Ø
       *        1HØ,1X,31HSEE SECTION 2 OF THE ALGORITHM.)            ØØØØ436Ø
        WRITE(6,1Ø)                                                   ØØØØ437Ø
        WRITE(6,11) (HR(K,K),K=1,N)                                   ØØØØ438Ø
        WRITE(6,11) (HI(K,K),K=1,N)                                   ØØØØ439Ø
        WRITE(6,12)                                                   ØØØØ44ØØ
        DO 1 K=1,NBLOCK                                               ØØØØ441Ø
        MULT=NDEL(K+1)-NDEL(K)                                        ØØØØ442Ø
        WRITE(6,13) NDEL(K+1),MULT,CSHTR(K),CSHTI(K)                  ØØØØ443Ø
      1 CONTINUE                                                      ØØØØ444Ø
        WRITE(6,14)                                                   ØØØØ445Ø
   14   FORMAT(1HØ,1X,34HC--IN STEP 6 OF THE ALGORITHM THE ,          ØØØØ446Ø
       *  5ØHSTRUCTURE OF EACH MULTIPLE EIGENVALUE IS COMPUTED.       ØØØØ447Ø
       * /1HØ,1X,42HFOR THAT REASON RDEFL SUCCESIVELY COMPUTES,       ØØØØ448Ø
       *  44H SINGULAR VALUE DECOMPOSITIONS. RDEFL PRINTS             ØØØØ449Ø
       * /1HØ,1X,39HTHE RESULTS BELOW(SEE ALSO COMMENTS IN ,          ØØØØ45ØØ
       *    35HRDEFL AND STEP 6 OF THE ALGORITHM).)                   ØØØØ451Ø
     1Ø  FORMAT(1HØ,1X,43HA--ENTER DECIDE WITH EIGENVALUES COMPUTED  ,  ØØØØ452Ø
       *          36HBY COMLR2 (STEP 1 OF THE ALGORITHM ))            ØØØØ453Ø
     11  FORMAT(1ØF13.9)                                             ØØØØ454Ø
   12      FORMAT(1X,4ØHB--GROUPINGS OF THE EIGENVALUES,COMPUTED,     ØØØØ455Ø
       *  27H BY STEP 3 OF THE ALGORITHM)                             ØØØØ456Ø
     13 FORMAT(12H DIVISION AT,I4, 7H MULT.=,I4, 8H CENTER=,2E2Ø.1Ø)  ØØØØ457Ø
        RETURN                                                        ØØØØ458Ø
        END                                                           ØØØØ459Ø


      SUBROUTINE JNF (NM,N,HR,HI,ZR,ZI,EVR,EVI,SUPD,NXT,NDEL,NDEFL,NDB, ØØØØ694Ø
     CNBLOCK,EIN,TOL,DELE,SM,IERR,ISTEP)                              ØØØØ695Ø
      REAL HR(NM,N),HI(NM,N),ZR(NM,N),ZI(NM,N),EVR(N),EVI(N),SUPD(N), ØØØØ696Ø
     CDELE(N),SM(N)                                                   ØØØØ697Ø
      INTEGER NXT(N),NDEL(N),NDEFL(N),NDB(N)                          ØØØØ698Ø
C                                                                     ØØØØ699Ø
C     ************************************************************    ØØØØ7ØØØ
C     *    THE FORMAL PARAMETER LIST                                 *  ØØØØ7Ø1Ø
```

```
C   *                                                                  ØØØØ7Ø2Ø
C   *    ON INPUT -                                                    ØØØØ7Ø3Ø
C   *                                                                  ØØØØ7Ø4Ø
C   *         NM    - MUST BE SET TO THE ROW DIMENSION OF THE TWO-     ØØØØ7Ø5Ø
C   *                 DIMENSIONAL ARRAY PARAMETERS AS DECLARED IN     ØØØØ7Ø6Ø
C   *                 THE CALLING PROGRAM                             ØØØØ7Ø7Ø
C   *         N     - IS THE ORDER OF THE MATRICES                    ØØØØ7Ø8Ø
C   *                                                                 ØØØØ7Ø9Ø
C   *         HR,HI - CONTAIN THE REAL AND IMAGINARY PARTS,RESPECTIVELY ØØØØ71ØØ
C   *                 OF THE COMPLEX MATRIX                           ØØØØ711Ø
C   *         EIN   - IS A TOLERANCE PARAMETER,CORRESPONDING TO       ØØØØ712Ø
C   *                 PERTURBATIONS OF HR,HI, AND IS USED IN THE      ØØØØ713Ø
C   *                 GROUPING OF NUMERICAL MULTIPLE EIGENVALUES      ØØØØ714Ø
C   *                                                                 ØØØØ715Ø
C   *         TOL   - IS A TOLERANCE PARAMETER USED IN THE CONSTRUCTION ØØØØ716Ø
C   *                 OF THE NILPOTENT MATRICES (IS USED IN RDEFL)    ØØØØ717Ø
C   *                                                                 ØØØØ718Ø
C   *          ISTEP  - RETURN FROM JNF AFTER STEP ISTEP(=1,2,3,4,5,6,7)ØØØØ719Ø
C   *                   FULL REDUCTION TO JORDAN FORM IF ISTEP .GE. 7 ØØØØ72ØØ
C   *                   OTHER CHOICES GIVE REDUCTION TO               ØØØØ721Ø
C   *                                                                 ØØØØ722Ø
C   *           1       UPPER TRIANGULAR FORM                         ØØØØ723Ø
C   *                                                                 ØØØØ724Ø
C   *           2       UPPER TRIANGULAR FORM WITH THE EIGEN-         ØØØØ725Ø
C   *                   VALUES SORTED SUCH THAT CLOSE EIGEN-          ØØØØ726Ø
C   *                   VALUES APPEAR IN ADJACENT POSITIONS           ØØØØ727Ø
C   *                                                                 ØØØØ728Ø
C   *           3       THE SAME AS 2,IN ADDITION GROUPING OF         ØØØØ729Ø
C   *                   CLOSE EIGENVALUES INTO BLOCKS CORRE-          ØØØØ73ØØ
C   *                   SPONDING TO NUMERICAL MULTIPLE EIGEN-         ØØØØ731Ø
C   *                   VALUES                                        ØØØØ732Ø
C   *                                                                 ØØØØ733Ø
C   *           4       BLOCK DIAGONAL UPPER TRIANGULAR FORM          ØØØØ734Ø
C   *                                                                 ØØØØ735Ø
C   *           5       BLOCK DIAGONAL UPPER TRIANGULAR FORM          ØØØØ736Ø
C   *                   AND THE INVARIANT SUBSPACES CORRESPONDING     ØØØØ737Ø
C   *                   TO THE DIAGONALS BLOCKS HAVE UNITARY BASES    ØØØØ738Ø
C   *                                                                 ØØØØ739Ø
C   *           6       THE STRUCTURE OF EACH DIAGONAL BLOCK IS       ØØØØ74ØØ
C   *                   DETERMINED                                    ØØØØ741Ø
C   *    ON OUTPUT -                                                  ØØØØ742Ø
C   *                                                                 ØØØØ743Ø
C   *  THE PARAMETERS CONTAIN USEFUL INFORMATION DEPENDING ON         ØØØØ744Ø
C   *  THE VALUE OF THE INPUT PARAMETER ISTEP.                        ØØØØ745Ø
C   *                                                                 ØØØØ746Ø
C   *  NOTATION - (IF ISTEP .GE. N )                                  ØØØØ747Ø
C   *  WHERE N IS 1,2,3,4,5,6 OR 7                                    ØØØØ748Ø
C   *                                                                 ØØØØ749Ø
C   *                                                                 ØØØØ75ØØ
C   *         HR,HI - HAVE BEEN DESTROYED                             ØØØØ751Ø
C   *                                                                 ØØØØ752Ø
C   *         IERR  - IS A CONVERGENCE PARAMETER SET BY COMLR2        ØØØØ753Ø
C   *                 Ø  FOR NORMAL RETURN                            ØØØØ754Ø
C   *                 J  IF THE J-TH EIGENVALUE HAS NOT BEEN DETERMINED ØØØØ755Ø
C   *                 AFTER 3Ø ITERATIONS                             ØØØØ756Ø
C   *                 IF AN ERROR EXIT IS MADE (IERR.NE.Ø),NONE       ØØØØ757Ø
C   *                 OF THE FOLLOWING PARAMETERS CONTAIN MEANINGFUL  ØØØØ758Ø
C   *                 RESULTS                                         ØØØØ759Ø
C   *                (IF ISTEP.GE.1)                                  ØØØØ76ØØ
C   *         EVR,EVI CONTAIN THE REAL AND IMAGINARY PARTS,RESPECTIVELY ØØØØ761Ø
C   *                 OF THE EIGENVALUES.                             ØØØØ762Ø
C   *                 IF ISTEP.GE.1 BUT .LT.5 THE EIGENVALUES ARE     ØØØØ763Ø
C   *                 COMPUTED BY COMLR2.                             ØØØØ764Ø
C   *                 IF ISTEP.GE.6, EVR,EVI CONTAIN THE MAIN DIAGONAL ØØØØ765Ø
C   *                 OF THE JORDAN MATRIX.                           ØØØØ766Ø
C   *                                                                 ØØØØ767Ø
C   *         NBLOCK- IS THE NUMBER OF NUMERICAL MULTIPLE EIGENVALUES ØØØØ768Ø
C   *                 (IF ISTEP.GE.3)                                 ØØØØ769Ø
C   *                                                                 ØØØØ77ØØ
C   *         NDEL  - INDICATES THE MULTIPLICITIES OF THE NUMERICAL   ØØØØ771Ø
C   *                 MULTIPLE EIGENVALUES                            ØØØØ772Ø
C   *                 (IF ISTEP.GE.3)                                 ØØØØ773Ø
C   *                 NDEL(I+1)-NDEL(I)= MULTIPLICITY OF EIGENVALUE   ØØØØ774Ø
C   *                                    I FOR I=1,...,NBLOCK         ØØØØ775Ø
C   *         NDEFL - INDICATES THE STRUCTURE OF HR,HI                ØØØØ776Ø
C   *                 (IF ISTEP.GE.6)                                 ØØØØ777Ø
```

```
C   *                                                                  ØØØØ778Ø
C   *         SUPD  - CONTAINS THE REAL SUPERDIAGONAL OF THE JORDAN    ØØØØ779Ø
C   *                 MATRIX (THE COUPLING ELEMENTS)                   ØØØØ78ØØ
C   *                 (IF ISTEP.GE.7)                                  ØØØØ781Ø
C   *                                                                  ØØØØ782Ø
C   *         NXT   - CONTAINS THE COLUMN INDICES OF SUPD              ØØØØ783Ø
C   *                 NXT(I)=J IMPLIES THAT THE VECTOR WITH COUPLING   ØØØØ784Ø
C   *                          ELEMENT SUPD(I) IS PLACED IN COLUMN     ØØØØ785Ø
C   *                          J OF ZR,ZI                              ØØØØ786Ø
C   *                 (IF ISTEP.GE.7)                                  ØØØØ787Ø
C   *         NDB   - CONTAINS POINTER REFERENCES                      ØØØØ788Ø
C   *                 NDB(I)=J MEANS THAT INFORMATION ABOUT THE        ØØØØ789Ø
C   *                          STRUCTURE OF THE NUMERICAL MULTIPLE     ØØØØ79ØØ
C   *                          EIGENVALUE I STARTS IN POSITION J       ØØØØ791Ø
C   *                          OF NDEFL                                ØØØØ792Ø
C   *                 (IF ISTEP.GE.6)                                  ØØØØ793Ø
C   *         ZR,ZI - CONTAIN THE REAL AND IMAGINARY PARTS,RESPECTIVELY ØØØØ794Ø
C   *                 OF THE ACCUMULATED TRANSFORMATIONS FROM STEP 1    ØØØØ795Ø
C   *                 TO 7 OF THE ALGORITHM.                           ØØØØ796Ø
C   *                 IF ISTEP.GE.7, ZR,ZI ARE THE EIGENVECTORS AND    ØØØØ797Ø
C   *                 THE PRINCIPAL VECTORS.                           ØØØØ798Ø
C   *                                                                  ØØØØ799Ø
C   *         DELE  - CONTAINS INFORMATION ABOUT DELETED ELEMENTS      ØØØØ8ØØØ
C   *                 DURING THE PROCESSES OF FINDING NILPOTENT MATRICESØØØØ8Ø1Ø
C   *                 DELE(I)= EUCLIDEAN NORM OF DELETED PART FOR BLOCK ØØØØ8Ø2Ø
C   *                          I FOR I=1,...,NBLOCK                    ØØØØ8Ø3Ø
C   *                 (IF ISTEP.GE.6)                                  ØØØØ8Ø4Ø
C   *                                                                  ØØØØ8Ø5Ø
C   *         SM    - CONTAINS ESTIMATES OF THE SPECTRAL PROJECTORS    ØØØØ8Ø6Ø
C   *                 CORRESPONDING TO DIFFERENT NUMERICAL MULTIPLE    ØØØØ8Ø7Ø
C   *                 EIGENVALUES                                      ØØØØ8Ø8Ø
C   *                 (IF ISTEP.GE.4)                                  ØØØØ8Ø9Ø
C   *                                                                  ØØØØ81ØØ
C   *    FOR COMPLETE DESCRIPTION SEE COMPUTATIONAL DETAILS IN         ØØØØ811Ø
C   *    THE TEXT                                                      ØØØØ812Ø
C   *                                                                  ØØØØ813Ø
```

ALGORITHM 561
Fortran Implementation of Heap Programs for Efficient Table Maintenance [Z]

DAVID K. KAHANER

Los Alamos Scientific Laboratory, University of California, and U.S. Department of Commerce National Bureau of Standards

Key Words and Phrases: heap, table maintenance
CR Categories: 3.74, 5.31
Language: Fortran

DESCRIPTION

1. Introduction

In many computational problems one has a collection of entities whose time history is being studied. The calculation proceeds by creating, destroying, and generally processing these entities, henceforth called nodes. Often the node to be processed is that one which is "most important" in some problem-dependent sense. Examples of such calculations occur in particle following for accelerator problems and adaptive quadrature, to name but two [3, 4, 6]. In such problems the time spent maintaining the nodes is significant, and it is important to implement this efficiently.

An organization of nodes based on "most important in, first out" (i.e., where every deletion removes the most important node) is called a priority queue [5]. The computer implementation of such queues can vary considerably. An unordered table of nodes is the simplest arrangement but requires an expensive search for the most important node. Maintaining an ordered table allows easy access to the most important node, but then insertions become slow. A "heap" [1] is a data structure that implements a priority queue in an efficient way, being equally fast for both insertions and deletions. Here we present a suite of Fortran programs of library quality for heaps. These programs are of modest length and can easily be included in other packages. Similar programs were used in [6] but they were not found to be general enough for library use.

In this paper we do not describe any of the detailed properties of heaps since these have been well documented [1, 7]. Rather we give only enough information about them so that the use of the programs can be quickly understood. The last section contains two simple examples.

In our implementation the work required to form the heap from N given nodes is $O(N)$, and the work required either to remove the most important and update or to insert and update is $O(\log_2 N)$. This data structure is appropriate if the current most important node might be processed and destroyed before other

Received 13 December 1977; revised 3 October 1979; accepted 11 December 1979.
This work was supported in part by the U.S. Department of Energy under Contract W-7405-ENG. 36.

Author's address: Division 713, National Bureau of Standards, Washington, DC 20234.

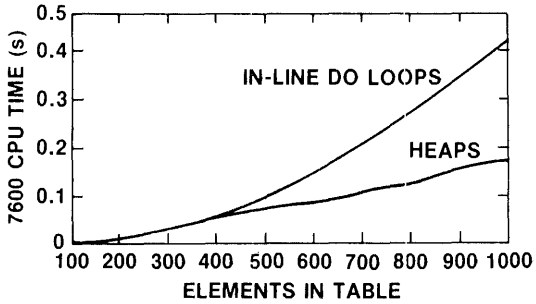


Fig. 1. Example of table management with heaps versus in-line DO loops.

nodes are created; the data structure is less appropriate if all the nodes are created before any are processed and destroyed.

The advantages of the programs presented in this paper over more familiar searching techniques are illustrated in Figure 1. Here we compare calls to the heap subroutines with in-line DO loops for the management of variously sized tables. This computation was selected to maximize the advantage of the in-line programming. (Specifically, example 1 of Section 3 was reprogrammed to use a DO loop at each stage to find the maximum element in an unordered table.) More typical problems should produce even more favorable time comparisons. For example, the production program MARLOWE, available from the Argonne Code Center, can have its execution time reduced 95 percent by using heaps instead of linear searches.

2. Using the Programs

We take a node to be one or more consecutive words of computer memory, with the data in these words defining the node. To investigate particle interactions (e.g., in an accelerator beam), each node could represent one particle, the nodal fields being position in phase space, energy, etc. For adaptive quadrature a node represents a portion of the region of integration; the fields in the node give the physical description of the region as well as local quadrature and error estimates. The most important node could be associated with the particle of largest energy or, for quadrature, the subregion with largest or smallest error, error per unit volume, that region nearest the origin, etc. To implement this variety, we require a Boolean function, HPFUN, on the nodes. Given two nodes A and B, this function returns .TRUE. if node A is "more important" than node B. The most important node is called TOP (i.e., it is the one for which HPFUN (TOP, X) = .TRUE. for any other node X).

Operations that can be performed on the heap, corresponding subroutines, and calling sequence parameters are shown in Table I. We also show (for completeness) the form of the Boolean HPFUN as well as an internally used subroutine HPGRO. A description of the parameters is given later.

The usual sequence of calls begins with HPINIT, perhaps followed by HPBLD. The main calculations usually require numerous calls to HPINS, HPACC, and HPDEL in various orders and combinations.

The package stores nodes into and removes them from a scratch area (DATA) provided by the user. The function HPFUN is an argument to the subroutines and thus may be changed from run to run or even during a single run. The latter situation might be useful during a calculation in which too many nodes are being generated.

The subroutine parameters are as follows:

NMAX Maximum number of nodes permitted, set by user.
NWDS Number of words per node, set by user.
DATA Real work array, at least NMAX * NWDS words, dimensioned by the user for node storage.

Table I. Heap Functions, Subroutines, and Calling Sequence Parameters

Function	Subroutine	Calling sequence parameters
Initialize package at start of each calculation	HPINIT	NMAX, N, T
Build a heap	HPBLD	NMAX, NWDS, DATA, N, T, HPFUN
Insert a node	HPINS	NMAX, NWDS, DATA, N, T, XNODE, HPFUN
Delete top node	HPDEL	NMAX, NWDS, DATA, N, T, HPFUN
Access a node (K = 1 gives top)	HPACC	NMAX, NWDS, DATA, N, T, XNODE, K
Determine the "more important" of two given nodes	HPFUN[a]	A, B, NWDS
Internally used subroutine	HPGRO	NMAX, NWDS, DATA, N, T, HPFUN, I

[a] Generic name—user may supply his own, but name must be EXTERNALed.

N Current number of nodes in the heap, set by the package.

T Integer work array dimensioned at least NMAX by the user, defined and used internally as a pointer.

XNODE Real array of NWDS words dimensioned by the user and used to store a single node. Thus to insert a node via HPINS, the user forms XNODE, i.e., defines its NWDS words and calls HPINS to copy these into the heap. To access a node with HPACC, the package copies the nodal data into XNODE (see examples).

HPFUN Name of a logical function with two real parameters and one integer parameter, of the form

```
LOGICAL FUNCTION HPFUN (A, B, NWDS)
INTEGER NWDS
REAL A(NWDS), B(NWDS)
```

The name of this function must be included in an EXTERNAL statement in the user calling program. When this function is called by the package, the arguments A and B will be two different nodes. The package provides two Boolean functions, LESS and GREATR, defined by

```
LESS = A(1) .LT. B(1)
GREATR = A(1) .GT. B(1)
```

These functions may be used when the characteristic on which the heap is organized appears in the first word of each node. The LESS and GREATR functions make the TOP smallest or largest, respectively. If the user wants more complicated functions, he must write them himself. For example, to select the node that is closest to the origin in the sense of the sum of squares of components, we can write

```
      LOGICAL FUNCTION L2 (A, B, NWDS)
      INTEGER NWDS, I
      REAL A (NWDS), B (NWDS), DA, DB
      DA = 0.0
      DB = 0.0
      DO 1 I = 1, NWDS
    1   DA = DA + A (I)**2
        DB = DB + B (I)**2
      L2 = DA .LT. DB
      RETURN
      END
```

K (In SUBROUTINE HPACC) the index of the node to be accessed; set by user. K = 1 corresponds to the TOP, and K must be in the range $1 \leq K \leq N \leq NMAX$. (Since the heap is only partially ordered, there is no way to predict the node produced by using this subroutine with $K > 1$. It is allowed as a convenience to those users who may wish to examine all the nodes without altering the heap structure.)

Note. If subroutines in the package are called with incorrect arguments, e.g., N > NMAX, the programs return without performing any calculations. No error diagnostics are given.

Operation Counts. The initialization program HPINIT and the infrequently used HPBLD require $O(N)$ operations. Inserting or deleting a node, HPINS or HPDEL, requires $O(\log_2 N)$ operations; accessing a node with HPACC requires $O(1)$ operations and has no effect upon the heap.

3. Examples of Use

Our first example is trivial. It has no application except to illustrate the use of the package. Nevertheless it forms the framework for the second example, adaptive quadrature, an important algorithm.

Example 1. A Simple Example of Heap Subroutines

A node is a single real number, between 0 and 1, generated at random. The TOP node in the heap is that one with largest numerical value. The calculation is as follows. Start with an initial node. While the number of nodes is less than 1000, print the value of the TOP node, remove it from the heap, and replace it by two others also selected at random. The complete Fortran program is as follows:

```
      REAL DATA (1000), XNODE
      INTEGER T(1000)
      EXTERNAL GREATR
      LOGICAL GREATR
C
      NMAX = 1000
      CALL HPINIT (NMAX, N, T)
      XNODE = RANNUM (0.)
      CALL HPINS (NMAX, 1, DATA, N, T, XNODE, GREATR)
C
    1 IF (N .GE. NMAX) STOP
        CALL HPACC (NMAX, 1, DATA, N, T, XNODE, 1)
        WRITE (6,100) XNODE
        CALL HPDEL (NMAX, 1, DATA, N, T, GREATR)
        XNODE = RANNUM (0.)
        CALL HPINS (NMAX, 1, DATA, N, T, XNODE, GREATR)
        XNODE = RANNUM (0.)
        CALL HPINS (NMAX, 1, DATA, N, T, XNODE, GREATR)
        GO TO 1
  100 FORMAT (E16.8)
      END
```

Example 2. Global Adaptive Quadrature

We start with an initial interval, a user-provided subroutine for evaluating the integrand, and $\epsilon > 0$. The algorithm generates subintervals by bisection and produces an integral estimate and error estimate for each. The subinterval subdivided next is that one having largest error estimate. The algorithm terminates where the sum of subinterval error estimates is less than ϵ. This is the prototype algorithm on which heaps were successfully used in [6]. A node is a real four-word array corresponding to an interval and has the form

$NODE_1$ = error estimate ≥ 0
$NODE_2$ = integral estimate

$NODE_3$ = left endpoint
$NODE_4$ = right endpoint

The heap is organized so that the TOP is the node with the largest error estimate. Thus the function GREATR provided in the package is used.

The main user subroutine is AGQ. Auxiliary calculations use the user-written programs:

QINIT	Initializes the first interval to a node.
QSDVD	Produces two half-interval nodes from each full interval.
QUAD	Provides integral estimate and error estimate for the user function on a given node.

In our sample program integral/error estimates are calculated by 2- and 3-point Gauss quadrature. This is, however, independent of any other aspect of the algorithm, and readers may substitute Newton–Cotes, Kronrod, etc. Our program has not taken advantage of the fact that for this problem a deletion is always followed by an insertion; the gain by doing this is very small. Furthermore the favorable timings of the heap programs relative to a simple list only appear after about 100 nodes, a situation which does not occur for the two problems in the example. On the other hand, for automatic integrators small gains in efficiency are not a major issue for problems generating such a small number of nodes. This same simple program organization can be used, however, for multidimensional adaptive quadrature after obvious changes in the definition and storage requirements for a node. For these problems large numbers of nodes can easily be generated. The program of Example 2 is a prototype of the library software that is now being developed in this area. For more complex applications of these ideas see [2] and [4].

REFERENCES

1. AHO, A., HOPCROFT, J., AND ULLMAN, J. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass., 1973.
2. GENTLEMAN, W.M. Row elimination for solving sparse linear systems and least squares problems. Presented at the Dundee Biennial Conference on Numerical Analysis, Univ. Dundee, Dundee, Scotland, July 1975.
3. GONNET, G.H., AND ROGERS, L.D. The algorithmic and complexity analysis of the heap as a data structure. Rep. CS-75-20, Comput. Sci. Dept., Univ. Waterloo, Waterloo, Ont., Canada, July 1975.
4. KAHANER, D.K., AND WELLS, M.B. An algorithm for *N*-dimensional adaptive quadrature using advanced programming techniques. Rep. LA-UR-76-2310, Los Alamos Sci. Lab., Los Alamos, N. Mex., Nov. 1976.
5. KNUTH, D. *The Art of Computer Programming I: Fundamental Algorithms.* Addison-Wesley, Reading, Mass., 1973.
6. MALCOLM, M.A., AND SIMPSON, R.B. Local vs. global strategies for adaptive quadrature. Rep. CS-74-07, Comput. Sci. Dept., Univ. Waterloo, Waterloo, Ont., Canada, May 1974.
7. WIRTH, N. *Algorithms + Data Structures = Programs.* Addison-Wesley, Reading, Mass., 1976.

ALGORITHM

```
      REAL WORK(1ØØØ),TNODE(4)                                          MAIN   1
      INTEGER IWORK(2ØØ)                                                MAIN   2
      EXTERNAL F,G,LEFT                                                 MAIN   3
      LOGICAL LEFT                                                      MAIN   4
C                                                                       MAIN   5
C          TEST DRIVER TO ILLUSTRATE THE USE OF HEAP SUBROUTINES FOR    MAIN   6
C            QUADRATURE.                                                MAIN   7
C          THIS PROGRAM CALLS THE ADAPTIVE QUADRATURE SUBROUTINE AGQ    MAIN   8
C            TO INTEGRATE THE TWO FUNCTIONS                             MAIN   9
C                  F(X)=EXP(X)                                          MAIN  1Ø
C            AND                                                        MAIN  11
C                  G(X)=1./SQRT(X)                                      MAIN  12
C            ON (Ø., 1.) TO AN ABSOLUTE ACCURACY OF 1.E-Ø5              MAIN  13
C          THE DEFINITIONS OF THE ARGUMENTS TO SUBROUTINE AGQ ARE GIVEN MAIN  14
C            IN THE COMMENTS FOR THAT SUBROUTINE.                       MAIN  15
C          FOR EACH OF THESE TWO PROBLEMS THIS TEST PROGRAM PRINTS      MAIN  16
```

```
C            Q = THE COMPUTED ESTIMATE OF THE INTEGRAL                    MAIN  17
C            E = THE COMPUTED ESTIMATE OF THE ERROR                       MAIN  18
C            N = THE NUMBER OF SUBINTERVALS IN THE FINAL PARTITION OF     MAIN  19
C                  THE INTERVAL (Ø., 1.)                                  MAIN  2Ø
C            IFLAG = A SUCCESS (Ø) OR FAILURE (1) INDICATOR.              MAIN  21
C                                                                         MAIN  22
      CALL AGQ(Ø.,1.,F,1.E-Ø5,Q,E,N,1ØØ,WORK,IWORK,IFLAG)                 MAIN  23
      WRITE (6,1Ø2)                                                       MAIN  24
      WRITE(6,1ØØ) Q,E,N,IFLAG                                            MAIN  25
      CALL AGQ(Ø.,1.,G,1.E-Ø5,Q,E,N,1ØØ,WORK,IWORK,IFLAG)                 MAIN  26
      WRITE(6,1ØØ) Q,E,N,IFLAG                                            MAIN  27
C                                                                         MAIN  28
C          AFTER THE SECOND PROBLEM WE REORGANIZE THE HEAP SO THAT THE    MAIN  29
C            ROOT INTERVAL IS THE LEFTMOST ONE, AND THEN PRINT THE        MAIN  3Ø
C            SUBINTERVALS USED LEFT TO RIGHT.                             MAIN  31
C                                                                         MAIN  32
      CALL HPBLD(1ØØ,4,WORK,N,IWORK,LEFT)                                 MAIN  33
      M=N                                                                 MAIN  34
      WRITE(6,1Ø3)                                                        MAIN  35
      DO 1 I=1,M                                                          MAIN  36
         CALL HPACC(1ØØ,4,WORK,N,IWORK,TNODE,1)                           MAIN  37
         WRITE(6,1Ø1) (TNODE(J), J=1,4)                                   MAIN  38
         CALL HPDEL(1ØØ,4,WORK,N,IWORK,LEFT)                              MAIN  39
    1 CONTINUE                                                            MAIN  4Ø
C                                                                         MAIN  41
  1ØØ FORMAT(1X,2E16.8,2I1Ø)                                              MAIN  42
  1Ø1 FORMAT( 1X,4E16.8)                                                  MAIN  43
  1Ø2 FORMAT(16H   QUAD. EST.   ,16H    ERR. EST.   ,1ØH NO. INTS ,1ØH SMAIN  44
     1UCCESS=Ø  )                                                         MAIN  45
  1Ø3 FORMAT(65H FINAL SUBINTERVALS...ERR EST, QUAD EST,  L. END PT,  RTMAIN  46
     1. END PT    )                                                       MAIN  47
      STOP                                                                MAIN  48
      END                                                                 MAIN  49


      LOGICAL FUNCTION LEFT(A,B,NCHAR)                                    LEFT   1
      DIMENSION A(1), B(1)                                                LEFT   2
C                                                                         LEFT   3
C          A TEST HEAP FUNCTION.                                          LEFT   4
C          THIS FUNCTION SETS LEFT = TRUE IF THE THIRD COMPONENT OF       LEFT   5
C            NODE A IS .LT. THE THIRD COMPONENT OF NODE B.                LEFT   6
C          FOR THE TEST PROBLEM THIS COMPONENT CONTAINS THE LEFT END-     LEFT   7
C            POINT OF THE INTERVAL DEFINED BY THE NODE.                   LEFT   8
C                                                                         LEFT   9
      LEFT= A(3) .LT. B(3)                                                LEFT  1Ø
      RETURN                                                              LEFT  11
      END                                                                 LEFT  12


      FUNCTION F(X)                                                       F      1
      F=EXP(X)                                                            F      2
      RETURN                                                              F      3
      END                                                                 F      4


      FUNCTION G(X)                                                       G      1
      G=1./SQRT(X)                                                        G      2
      RETURN                                                              G      3
      END                                                                 G      4


      SUBROUTINE AGQ(A,B,F,EPS,Q,E,N,NMAX,XNODES,T,IFLAG)                 AGQ    1
C                                                                         AGQ    2
C          1-D ADAPTIVE QUADRATURE-EXAMPLE USING HEAPS                    AGQ    3
C   INPUT                                                                 AGQ    4
C        A, B =  LIMITS OF INTEGRATION  A.LT.B                            AGQ    5
C        F = NAME OF USER SUPPLIED INTEGRAND FUNCTION, FUNCTION F(X)      AGQ    6
C        EPS = REQUESTED ABSOLUTE ACCURACY OF QUADRATURE                  AGQ    7
C        NMAX = MAX NUMBER OF NODES ALLOWED                               AGQ    8
C        XNODES = ARRAY FOR NODE STORAGE, DIMENSIONED AT LEAST 4*NMAX     AGQ    9
C        T = INTEGER ARRAY USED INTERNALLY, DIMENSIONED AT LEAST NMAX     AGQ   1Ø
C   OUTPUT                                                                AGQ   11
C        Q = QUADRATURE ESTIMATE                                          AGQ   12
C        E = ERROR ESTIMATE                                               AGQ   13
C        N = NUMBER OF NODES REQUIRED FOR THE QUADRATURE                  AGQ   14
```

```
C        IFLAG = ERROR INDICATOR                                          AGQ   15
C          = Ø PROGRAM APPARENTLY SUCCESSFUL                              AGQ   16
C          = 1 NODE STORAGE EXCEEDED, CALCULATION TERMINATED              AGQ   17
C                                                                         AGQ   18
C        EACH NODE IN THE HEAP REPRESENTS A SUBINTERVAL AND IS            AGQ   19
C          A FOUR WORD ARRAY.                                             AGQ   2Ø
C            NODE(1) = ERROR ESTIMATE                                     AGQ   21
C            NODE(2) = QUADRATURE ESTIMATE                                AGQ   22
C            NODE(3) = LEFT INTERVAL ENDPOINT                             AGQ   23
C            NODE(4) = RIGHT INTERVAL ENDPOINT                            AGQ   24
C                                                                         AGQ   25
      EXTERNAL GREATR,F                                                   AGQ   26
      LOGICAL GREATR                                                      AGQ   27
      REAL TNODE(4),N1(4),N2(4),XNODES(1)                                 AGQ   28
      INTEGER T(1)                                                        AGQ   29
C                                                                         AGQ   3Ø
      CALL HPINIT(NMAX,N,T)                                               AGQ   31
      CALL QINIT(TNODE,A,B)                                               AGQ   32
      E=TNODE(1)                                                          AGQ   33
      Q=TNODE(2)                                                          AGQ   34
C                                                                         AGQ   35
C        INSERT INITIAL NODE                                              AGQ   36
C                                                                         AGQ   37
      CALL HPINS(NMAX,4,XNODES,N,T,TNODE,GREATR)                          AGQ   38
C                                                                         AGQ   39
    1 IF(E .LE. EPS) RETURN                                               AGQ   4Ø
        CALL HPACC(NMAX,4,XNODES,N,T,TNODE,1)                             AGQ   41
        CALL HPDEL(NMAX,4,XNODES,N,T,GREATR)                              AGQ   42
        CALL QSDVD(TNODE,N1,N2,F)                                         AGQ   43
        CALL HPINS(NMAX,4,XNODES,N,T,N1,GREATR)                           AGQ   44
        CALL HPINS(NMAX,4,XNODES,N,T,N2,GREATR)                           AGQ   45
        E=E-TNODE(1)+N1(1)+N2(1)                                          AGQ   46
        Q=Q-TNODE(2)+N1(2)+N2(2)                                          AGQ   47
        IFLAG=1                                                           AGQ   48
        IF(N .GE. NMAX-1) RETURN                                          AGQ   49
        IFLAG=Ø                                                           AGQ   5Ø
        GO TO 1                                                           AGQ   51
      END                                                                 AGQ   52


      SUBROUTINE QINIT(NODE,A,B)                                          QINIT  1
C                                                                         QINIT  2
C          INITIALIZES THE FIRST INTERVAL TO A NODE BY SETTING THE        QINIT  3
C          ERROR ESTIMATE AND QUADRATURE ESTIMATE TO 1ØØØ. AND Ø. RESP.   QINIT  4
C                                                                         QINIT  5
      REAL NODE(4)                                                        QINIT  6
      NODE(1)=1ØØØ.                                                       QINIT  7
      NODE(2)=Ø.Ø                                                         QINIT  8
      NODE(3)=A                                                           QINIT  9
      NODE(4)=B                                                           QINIT 1Ø
      RETURN                                                              QINIT 11
      END                                                                 QINIT 12


      SUBROUTINE QSDVD(NODE,R,L,F)                                        QSDVD  1
C                                                                         QSDVD  2
C          SUBDIVIDES A NODE INTO TWO EQUAL HALVES.                       QSDVD  3
C          THE ENDPOINTS OF THE NEW NODES ARE COMPUTED HERE.  THE ERROR   QSDVD  4
C          AND QUADRATURE ESTIMATES FOR THE LEFT (L) AND RIGHT (R)        QSDVD  5
C          HALVES ARE COMPUTED BY THE SUBROUTINE QUAD.                    QSDVD  6
C                                                                         QSDVD  7
      EXTERNAL F                                                          QSDVD  8
      REAL NODE(4),R(4),L(4)                                              QSDVD  9
      L(3)=NODE(3)                                                        QSDVD 1Ø
      R(4)=NODE(4)                                                        QSDVD 11
      L(4)=(L(3)+R(4))/2.                                                 QSDVD 12
      R(3)=L(4)                                                           QSDVD 13
      CALL QUAD(L,F)                                                      QSDVD 14
      CALL QUAD(R,F)                                                      QSDVD 15
      RETURN                                                              QSDVD 16
      END                                                                 QSDVD 17


      SUBROUTINE QUAD(NODE,F)                                             QUAD   1
C                                                                         QUAD   2
C          ESTIMATES INTEGRAL OF F ON NODE USING 2 AND 3 POINT GAUSS.     QUAD   3
```

```
C          THE 3 POINT FORMULA GIVES ESTIMATE AND DIFFERENCE OF 2 AND 3 QUAD   4
C          GIVES ERROR ESTIMATE                                         QUAD   5
C                                                                       QUAD   6
      REAL NODE(4)                                                      QUAD   7
      XL=(NODE(4)-NODE(3))/2.                                           QUAD   8
      XM=(NODE(4)+NODE(3))/2.                                           QUAD   9
      Q2=XL*(F(XL*(-.57735Ø2692)+XM)+F(XL*(.57735Ø2692)+XM))            QUAD  1Ø
      Q3=XL*((F(XL*(-.7745966692)+XM)+F(XL*(.7745966692)+XM))/1.8+      QUAD  11
     1 F(XM)/1.125)                                                     QUAD  12
      NODE(1)=ABS(Q2-Q3)                                                QUAD  13
      NODE(2)=Q3                                                        QUAD  14
      RETURN                                                            QUAD  15
      END                                                               QUAD  16


C        H E A P  PACKAGE                                               HPINIT 1
C          A COLLECTION OF PROGRAMS WHICH MAINTAIN A HEAP DATA          HPINIT 2
C          STRUCTURE.  BY CALLING THESE SUBROUTINES IT IS POSSIBLE TO   HPINIT 3
C          INSERT, DELETE, AND ACCESS AN EXISTING HEAP OR TO BUILD A    HPINIT 4
C          NEW HEAP FROM AN UNORDERED COLLECTION OF NODES. THE HEAP     HPINIT 5
C          FUNCTION IS AN ARGUMENT TO THE SUBROUTINES ALLOWING VERY     HPINIT 6
C          GENERAL ORGANIZATIONS.                                       HPINIT 7
C            THE USER MUST DECIDE ON THE MAXIMUM NUMBER OF NODES        HPINIT 8
C          ALLOWED AND DIMENSION THE REAL ARRAY DATA AND THE INTEGER    HPINIT 9
C          ARRAY T USED INTERNALLY BY THE PACKAGE.  THESE VARIABLES ARE HPINIT1Ø
C          THEN PASSED THROUGH THE CALL SEQUENCE BETWEEN THE HEAP       HPINIT11
C          PROGRAMS BUT ARE NOT IN GENERAL ACCESSED BY THE USER.  HE    HPINIT12
C          MUST ALSO PROVIDE A HEAP FUNCTION WHOSE NAME MUST BE INCLUD- HPINIT13
C          ED IN AN EXTERNAL STATEMENT IN THE USER PROGRAM WHICH CALLS  HPINIT14
C          THE HEAP SUBROUTINES.  TWO SIMPLE HEAP FUNCTIONS ARE         HPINIT15
C          PROVIDED WITH THE PACKAGE.                                   HPINIT16
C                                                                       HPINIT17
      SUBROUTINE HPINIT(NMAX,N,T)                                       HPINIT18
C                                                                       HPINIT19
C   PURPOSE                                                             HPINIT2Ø
C         THIS ROUTINE INITIALIZES THE HEAP PROGRAMS.                   HPINIT21
C         IT IS CALLED ONCE AT THE START OF EACH NEW CALCULATION        HPINIT22
C   INPUT                                                               HPINIT23
C         NMAX = MAXIMUM NUMBER OF NODES ALLOWED BY USER.               HPINIT24
C   OUTPUT                                                              HPINIT25
C         N = CURRENT NUMBER OF NODES IN HEAP = Ø.                      HPINIT26
C         T = INTEGER ARRAY OF POINTERS TO POTENTIAL HEAP NODES.        HPINIT27
C                                                                       HPINIT28
      INTEGER T(1)                                                      HPINIT29
      DO 1 I = 1, NMAX                                                  HPINIT3Ø
    1    T(I)=I                                                         HPINIT31
      N = Ø                                                             HPINIT32
      RETURN                                                            HPINIT33
      END                                                               HPINIT34


      SUBROUTINE HPINS(NMAX,NCHAR,DATA,N,T,NODE,HPFUN)                  HPINS  1
C                                                                       HPINS  2
C   PURPOSE                                                             HPINS  3
C         THIS ROUTINE INSERTS A NODE INTO AN ALREADY EXISTING HEAP.    HPINS  4
C             THE RESULTING TREE IS RE-HEAPED.                          HPINS  5
C                                                                       HPINS  6
C   INPUT                                                               HPINS  7
C         NMAX = MAXIMUM NUMBER OF NODES ALLOWED BY USER.               HPINS  8
C         NCHAR = NUMBER OF WORDS PER NODE.                             HPINS  9
C         DATA = WORK AREA FOR STORING NODES.                           HPINS 1Ø
C         N = CURRENT NUMBER OF NODES IN THE TREE.                      HPINS 11
C         T = INTEGER ARRAY OF POINTERS TO HEAP NODES.                  HPINS 12
C         NODE = A REAL ARRAY, NCHAR WORDS LONG, WHICH                  HPINS 13
C                CONTAINS THE NODAL INFORMATION TO BE INSERTED.         HPINS 14
C         HPFUN = NAME OF USER WRITTEN FUNCTION TO DETERMINE            HPINS 15
C                 THE ROOT NODE.                                        HPINS 16
C   OUTPUT                                                              HPINS 17
C         DATA = WORK AREA WITH NEW NODE INSERTED.                      HPINS 18
C         N = UPDATED NUMBER OF NODES.                                  HPINS 19
C         T = UPDATED INTEGER POINTER ARRAY.                            HPINS 2Ø
C                                                                       HPINS 21
      DIMENSION T(1),NODE(1),DATA(1)                                    HPINS 22
      REAL NODE                                                         HPINS 23
      INTEGER T                                                         HPINS 24
      LOGICAL HPFUN                                                     HPINS 25
```

```
      IF(N .EQ. NMAX) RETURN                                          HPINS 26
      N=N+1                                                           HPINS 27
      J= (T(N)-1)*NCHAR                                               HPINS 28
      DO 1 I= 1,NCHAR                                                 HPINS 29
         IPJ=I+J                                                      HPINS 30
    1    DATA(IPJ) = NODE(I)                                          HPINS 31
      J=N                                                             HPINS 32
    2 CONTINUE                                                        HPINS 33
      IF(J .EQ. 1) RETURN                                             HPINS 34
      JT=T(J)                                                         HPINS 35
      J2=J/2                                                          HPINS 36
      JT2=T(J2)                                                       HPINS 37
      JL=(JT2-1)*NCHAR+1                                              HPINS 38
      JR=(JT-1)*NCHAR+1                                               HPINS 39
      IF(HPFUN(DATA(JL),DATA(JR),NCHAR)) RETURN                       HPINS 40
      T(J2)=T(J)                                                      HPINS 41
      T(J)=JT2                                                        HPINS 42
      J=J/2                                                           HPINS 43
      GO TO 2                                                         HPINS 44
      END                                                             HPINS 45


      SUBROUTINE HPBLD(NMAX,NCHAR,DATA,NELTS,T,HPFUN)                 HPBLD  1
C                                                                     HPBLD  2
C   PURPOSE                                                           HPBLD  3
C          BUILDS A HEAP, IN  T , FROM AN ARRAY OF  NELTS  ELEMENTS   HPBLD  4
C            IN DATA, WHICH ARE SPACED NCHAR APART.                   HPBLD  5
C            AT CONCLUSION OF CALCULATION  THE ROOT SATISFIES         HPBLD  6
C            HPFUN(ROOT,SON) = .TRUE. FOR ANY SON.                    HPBLD  7
C          USES SUBROUTINE HPGRO BY FEEDING IT ONE ELEMENT OF         HPBLD  8
C          THE ARRAY AT A TIME.                                       HPBLD  9
C                                                                     HPBLD 10
C   INPUT                                                             HPBLD 11
C         NMAX = MAXIMUN NUMBER OF NODES ALLOWED BY USER.             HPBLD 12
C         NCHAR = NUMBER OF WORDS PER NODE.                           HPBLD 13
C         DATA = WORK AREA IN WHICH THE NODES ARE STORED.             HPBLD 14
C         NELTS = CURRENT NUMBER OF NODES.                            HPBLD 15
C         T = INTEGER ARRAY OF POINTERS TO HEAP NODES.                HPBLD 16
C         HPFUN = NAME OF USER WRITTEN FUNCTION TO DETERMINE ROOT NODE. HPBLD 17
C   OUTPUT                                                            HPBLD 18
C         DATA = WORK AREA IN WHICH THE NODES ARE STORED.             HPBLD 19
C         T = INTEGER ARRAY OF POINTERS TO HEAP NODES.                HPBLD 20
C              IN PARTICULAR T(1) POINTS TO THE ROOT.                 HPBLD 21
C                                                                     HPBLD 22
      EXTERNAL HPFUN                                                  HPBLD 23
      LOGICAL HPFUN                                                   HPBLD 24
      DIMENSION DATA(1)                                               HPBLD 25
      INTEGER T(1)                                                    HPBLD 26
      IF(NMAX .LT. NELTS) RETURN                                      HPBLD 27
      INDEX = NELTS/2                                                 HPBLD 28
    1 CONTINUE                                                        HPBLD 29
      IF( INDEX .EQ. 0) RETURN                                        HPBLD 30
         CALL HPGRO(NMAX,NCHAR,DATA,NELTS,T,HPFUN,INDEX)              HPBLD 31
         INDEX = INDEX-1                                              HPBLD 32
         GO TO 1                                                      HPBLD 33
      END                                                             HPBLD 34


      SUBROUTINE HPDEL(NMAX,NCHAR,DATA,NCELLS,T,HPFUN)                HPDEL  1
C                                                                     HPDEL  2
C   PURPOSE                                                           HPDEL  3
C          DELETE ROOT ELEMENT OF HEAP.  RESULTING TREE IS REHEAPED.  HPDEL  4
C   INPUT                                                             HPDEL  5
C         NMAX = MAXIMUN NUMBER OF NODES ALLOWED BY USER.             HPDEL  6
C         NCHAR = NUMBER OF WORDS PER NODE.                           HPDEL  7
C         DATA = WORK AREA IN WHICH THE NODES ARE STORED.             HPDEL  8
C        NCELLS = CURRENT NUMBER OF NODES.                            HPDEL  9
C        T = INTEGER ARRAY OF POINTERS TO NODES.                      HPDEL 10
C         HPFUN = NAME OF USER WRITTEN FUNCTION TO DETERMINE ROOT NODE. HPDEL 11
C   OUTPUT                                                            HPDEL 12
C         NCELLS = UPDATED NUMBER OF NODES.                           HPDEL 13
C         T = UPDATED INTEGER POINTER ARRAY TO NODES.                 HPDEL 14
C                                                                     HPDEL 15
      EXTERNAL HPFUN                                                  HPDEL 16
      LOGICAL HPFUN                                                   HPDEL 17
```

```
      DIMENSION DATA(1)                                                 HPDEL 18
      INTEGER T(1)                                                      HPDEL 19
      IF(NCELLS .EQ. Ø) RETURN                                          HPDEL 2Ø
      JUNK=T(1)                                                         HPDEL 21
      T(1)=T(NCELLS)                                                    HPDEL 22
      T(NCELLS)=JUNK                                                    HPDEL 23
      NCELLS=NCELLS-1                                                   HPDEL 24
      CALL HPGRO(NMAX,NCHAR,DATA,NCELLS,T,HPFUN,1)                      HPDEL 25
      RETURN                                                            HPDEL 26
      END                                                               HPDEL 27


      SUBROUTINE HPACC(NMAX,NCHAR,DATA,N,T,NODE,K)                      HPACC  1
C                                                                       HPACC  2
C   PURPOSE                                                             HPACC  3
C         TO ACCESS THE K-TH NODE OF THE HEAP,                          HPACC  4
C         1 .LE. K .LE. N .LE. NMAX                                     HPACC  5
C   INPUT                                                               HPACC  6
C        NMAX = MAXIMUM NUMBER OF NODES ALLOWED BY USER.                HPACC  7
C        DATA = WORK AREA FOR STORING NODES.                            HPACC  8
C        N = CURRENT NUMBER OF NODES IN THE HEAP.                       HPACC  9
C        T = INTEGER ARRAY OF POINTERS TO HEAP NODES.                   HPACC 1Ø
C        NODE = A REAL ARRAY, NCHAR WORDS LONG, IN WHICH NODAL IN-      HPACC 11
C          FORMATION WILL BE INSERTED.                                  HPACC 12
C        K = THE INDEX OF THE NODE TO BE FOUND AND INSERTED INTO NODE.  HPACC 13
C                                                                       HPACC 14
C   OUTPUT                                                              HPACC 15
C        NODE =  A REAL ARRAY.    CONTAINS IN NODE(1),...,NODE(NCHAR)   HPACC 16
C          THE ELEMENTS OF THE K-TH NODE.                               HPACC 17
C                                                                       HPACC 18
      REAL DATA(1), NODE(1)                                             HPACC 19
      INTEGER T(1)                                                      HPACC 2Ø
      IF (K .LT. 1 .OR. K .GT. N .OR. N .GT. NMAX) RETURN               HPACC 21
      J=(T(K)-1)*NCHAR                                                  HPACC 22
      DO 1 I=1,NCHAR                                                    HPACC 23
         IPJ=I+J                                                        HPACC 24
    1    NODE(I)=DATA(IPJ)                                             HPACC 25
      RETURN                                                            HPACC 26
      END                                                               HPACC 27


      LOGICAL FUNCTION GREATR(A,B,NCHAR)                                GREATR 1
       REAL A(1),B(1)                                                   GREATR 2
      GREATR= A(1) .GT. B(1)                                            GREATR 3
      RETURN                                                            GREATR 4
      END                                                               GREATR 5


      LOGICAL FUNCTION LESS(A,B,NCHAR)                                  LESS   1
      REAL A(1),B(1)                                                    LESS   2
      LESS= A(1) .LT. B(1)                                              LESS   3
      RETURN                                                            LESS   4
      END                                                               LESS   5


      SUBROUTINE HPGRO(NMAX,KD,DATA,NELTS,T,HPFUN,KTEMP)                HPGRO  1
C                                                                       HPGRO  2
C   PURPOSE                                                             HPGRO  3
C         FORMS A HEAP OUT OF A TREE. USED PRIVATELY BY HPBLD.          HPGRO  4
C         THE ROOT OF THE TREE IS STORED IN LOCATION T(KTEMP).          HPGRO  5
C         FIRST SON IS IN LOCATION T(2KTEMP), NEXT SON                  HPGRO  6
C         IS IN LOCATION T(2KTEMP+1).                                   HPGRO  7
C         THIS PROGRAM ASSUMES EACH BRANCH OF THE TREE IS A HEAP.       HPGRO  8
C                                                                       HPGRO  9
      DIMENSION T(1),DATA(1)                                            HPGRO 1Ø
      INTEGER T                                                         HPGRO 11
      LOGICAL HPFUN                                                     HPGRO 12
      IF(NELTS .GT. NMAX) RETURN                                        HPGRO 13
C                                                                       HPGRO 14
      K=KTEMP                                                           HPGRO 15
    1 I=2*K                                                             HPGRO 16
C                                                                       HPGRO 17
C         TEST IF ELEMENT IN I TH POSITION IS A LEAF.                   HPGRO 18
C                                                                       HPGRO 19
      IF( I .GT. NELTS ) RETURN                                         HPGRO 2Ø
```

```
C                                                                       HPFRO 21
C          IF THERE IS MORE THAN ONE SON, FIND WHICH SON IS SMALLEST.   HPGRO 22
C                                                                       HPGRO 23
      IF( I .EQ. NELTS ) GO TO 2                                        HPGRO 24
      ITEMP=T(I)                                                        HPGRO 25
      ITP1=T(I+1)                                                       HPGRO 26
      IL=(ITP1-1)*KD+1                                                  HPGRO 27
      IR=(ITEMP-1)*KD+1                                                 HPGRO 28
      IF(HPFUN(DATA(IL),DATA(IR),KD)) I=I+1                             HPGRO 29
C                                                                       HPGRO 3Ø
C          IF A SON IS LARGER THAN FATHER, INTERCHANGE                  HPGRO 31
C          THIS DESTROYS HEAP PROPERTY, SO MUST RE-HEAP REMAINING       HPGRO 32
C          ELEMENTS                                                     HPGRO 33
C                                                                       HPGRO 34
    2 CONTINUE                                                          HPGRO 35
      KT=T(K)                                                           HPGRO 36
      ITEMP=T(I)                                                        HPGRO 37
      IL=(KT-1)*KD+1                                                    HPGRO 38
      IR=(ITEMP-1)*KD+1                                                 HPGRO 39
      IF(HPFUN(DATA(IL),DATA(IR),KD)) RETURN                            HPGRO 4Ø
         ITEMP=T(I)                                                     HPGRO 41
         T(I)=T(K)                                                      HPGRO 42
         T(K)=ITEMP                                                     HPGRO 43
         K=I                                                            HPGRO 44
      GO TO 1                                                           HPGRO 45
      END                                                               HPGRO 46
```

ALGORITHM 562
Shortest Path Lengths [H]

U. PAPE
Technische Universität Berlin

Key Words and Phrases: shortest path, shortest route problem
CR Categories: 5.32
Language: Fortran IV

DESCRIPTION

This algorithm finds the shortest path lengths from a specific node to all other nodes in a network. It is a modification of Moore's algorithm [4], originally due to d'Esopo as reported by Pollack and Wiebenson [7] and refined by Pape [5, 6]. The algorithm does not determine the shortest paths. If a shortest path is to be determined, this algorithm may be combined with any algorithm that traces the shortest path from the root node to any other node from the list of predecessors of nodes. As the calculation of the path lengths is the most time-consuming part of the whole process, we confine our attention to this problem.

The main idea of the given algorithm centers around the "status" of a newfound successor node j. If j has not yet been reached by the process, it is entered at the end of the successor list; if it is currently in the list, no new entry is made, but if it has already been processed and removed from the list, it is entered at the top of the list so that it will be processed next. The current status of each node is recorded by the values of the list which are stored as linked deque (double-ended queue). The steps involved are as follows:

```
top element of the deque := j;
mj := ∞; mj_j := 0;
while deque ≠ ∅ do
begin i := top element of the deque;
  remove top element;
  for all successors k of i do
  begin mjk := mj_i + d_ik
    if mjk < mj_k then
    begin mj_k := mjk; wj_k := i;
      if k has not yet been reached then enter k at the end of the deque;
      if k has already been processed then enter k at the top of the deque
    end
  end
end;
```

The great advantage of the above-mentioned technique is that errors in minimal distances are often corrected as soon as they are detected and not

Received 15 May 1976; revised 17 December 1979; accepted 26 February 1980.

Author's address: Technische Universität Berlin, Fachbereich 20, Lehreinheit Angewandte Elektronische Datenverarbeitung, Berlin 10, West Germany.

allowed to progress further. In practice this leads to a considerable reduction in CPU time, particularly for grid networks.

The additional bookkeeping work required costs less than the savings from the early processing of the nodes. This follows from a comparison of this technique with a more obvious strategy, where the deque is handled as a FIFO list (see [6]).

The network N is represented by its forward star structure (see [3]) which is also used as input to the subroutine. This input consists of three arrays: the array KF, containing information about where the data for each node begin in the other two arrays; the array FList, containing the numbers of the nodes which can be reached directly from each node; and the integer array DFList, containing the distances to each of these nodes.

The algorithm has been coded and tested in Algol 60, Algol-W, and Fortran IV for an IBM 370/158 (Berlin), an ICL 1907 (Braunschweig), and a CDC 6600 (Austin). It has been tested on different networks, real transportation/road networks and random road networks, constructed by network generators developed at Berlin and Austin. Independently, a variant of this algorithm has been tested at the Institute for Transportation Studies at the University of Leeds [8]. It has been compared with several versions of Dijkstra's algorithm [2] (with different binary tree structures for the nodes to be discussed), particularly Dial's algorithm [1]. The investigations have shown that the algorithm, based on d'Esopo's idea, is generally the most efficient in terms of CPU times (and core storage) for a large variety of networks. The code is highly insensitive to the magnitude of the distance measures.

REFERENCES

1. DIAL, R.B. Algorithm 360: Shortest-path forest with topological ordering. *Commun. ACM 12*, 11 (Nov. 1969), 632–633.
2. DIJKSTRA, E.W. Note on two problems in connection with graphs. *Numer. Math. 1* (1959), 269.
3. GILSINN, J., AND WITZGALL, C. A performance comparison of labeling algorithms for calculating shortest path trees. Tech. Note 772, National Bureau of Standards, May 1973.
4. MOORE, E.F. The shortest path through a maze. In Proc. Int. Symp. Theory of Switching, Harvard Univ., 1957, part 2.
5. PAPE, U. Implementation and efficiency of Moore-Algorithms for the shortest path problem. *Math. Progr. 7* (1974), 212–222.
6. PAPE, U. Zur Implementierung und Wirtschaftlichkeit von MOORE-Algorithmen zur Bestimmung kürzester Weglängen in einem Netzwerk. Tech. Rep. 73-06, Fachbereich 20, Tech. Univ., Berlin, Germany, 1973.
7. POLLACK, M., AND WIEBENSON, D. Solutions of the shortest route problem: a review *Oper. Res.* 8 (1960), 225.
8. VAN VLIET, D. Improved shortest path algorithms for transportation networks. Tech. Rep. ICL 1906A, Univ. Leeds, England, 1975; *Transport. Res. 12* (1978), 7–20.

ALGORITHM

```
2Ø
5     2     4     3     5     6
      8     8     8     9    14
6    1Ø     7     6     1    14    15
      9     9    1Ø     8    14    14
4     6     1    11     8
      8     8     8    1Ø
4     7    12     1     9
      9     8     8     9
5     1     9     8    13    18
      9     9     8     9    14
3    14     2     3
      9    1Ø     8
3    16     2     4
      8     9     9
2     3     5
     1Ø     8
5     4     5    18     1    12
      9     9     9    14    14
3    15    14     2
```

```
          8       8       9
     2    3      19
          8      2Ø
     2    4       7
          8      14
     2    5      17
          9       9
     4   1Ø      19       6       2
          8      1Ø       9      14
     2   1Ø       2
          8      14
     1    7
          8
     2   13      2Ø
          9       8
     2    9       5
          9      14
     2   14      11
         1Ø      2Ø
     1   17
          8
1
1
2
3
4
5
6
7
8
9
1Ø
11
12
13
14
15
16
17
18
19
2Ø
 Ø
 3
2Ø
15
1Ø
 5
 6
 9
 Ø
11
 2
 Ø
2Ø
16
 8
 4
 2
 1
 Ø
 Ø
```

```
      INTEGER FLIST(2ØØ),DFLIST(2ØØ),KF(51),MJ(5Ø),WJ(5Ø),W(5Ø),H(5Ø),  PAPØØØØ1
     +        IN,INPUT,OUTP,TEMP(1Ø)                                    PAPØØØØ2
C                                                                       PAPØØØØ3
C     THIS PROGRAM CALCULATES THE SHORTEST PATH LENGTHS FROM A SPECIFIC PAPØØØØ4
C     NODE (JØ) TO ALL OTHER NODES IN A NETWORK  AND THE SHORTEST PATHS PAPØØØØ5
C     BETWEEN THIS NODE (JØ) AND OTHER NODES (KE).                      PAPØØØØ6
C                                                                       PAPØØØØ7
C     FLIST(NUMBER OF EDGES)                                            PAPØØØØ8
C     DFLIST(NUMBER OF EDGES)                                           PAPØØØØ9
C     KF(NUMBER OF NODES+1)                                             PAPØØØ1Ø
C                                                                       PAPØØØ11
C     FLIST, DFLIST, AND KF REPRESENT THE NETWORK                       PAPØØØ12
```

```
C                                                                       PAPØØØ13
C     MJ(NUMBER OF NODES)                                               PAPØØØ14
C     WJ(NUMBER OF NODES)                                               PAPØØØ15
C     W(NUMBER OF NODES)                                                PAPØØØ16
C     H(NUMBER OF NODES)                                                PAPØØØ17
C                                                                       PAPØØØ18
      DATA INF,IN,INPUT,OUTP /1ØØØØØØ,4,5,6/                           PAPØØØ19
C                                                                       PAPØØØ2Ø
C     IN            CHANNEL FOR INPUT JØ                              PAPØØØ21
C     INPUT         CHANNEL FOR INPUT NETWORK                         PAPØØØ22
C     OUTP          CHANNEL FOR OUTPUT                                PAPØØØ23
C                                                                       PAPØØØ24
C     INPUT OF THE NUMBER OF NODES N                                    PAPØØØ25
C                                                                       PAPØØØ26
    1 READ(INPUT,1ØØ) N                                                 PAPØØØ27
      K=Ø                                                               PAPØØØ28
      KF(1)=Ø                                                           PAPØØØ29
C                                                                       PAPØØØ3Ø
C     INPUT OF THE NETWORK                                              PAPØØØ31
C                                                                       PAPØØØ32
      DO 4 I=1,N                                                        PAPØØØ33
      READ(INPUT,1ØØ) IT,(TEMP(J),J=1,IT)                               PAPØØØ34
      IF (IT.EQ.Ø) GO TO 3                                              PAPØØØ35
      JA=K+1                                                            PAPØØØ36
      JB=K+IT                                                           PAPØØØ37
      JJ=1 .                                                            PAPØØØ38
      DO 2 J=JA,JB                                                      PAPØØØ39
      FLIST(J)=TEMP(JJ)                                                 PAPØØØ4Ø
    2 JJ=JJ+1                                                           PAPØØØ41
C                                                                       PAPØØØ42
      READ(INPUT,1ØØ) IDUMMY,(DFLIST(J),J=JA,JB)                        PAPØØØ43
      K=K+IT                                                            PAPØØØ44
    3 KF(I+1)=K                                                         PAPØØØ45
    4 CONTINUE                                                          PAPØØØ46
C                                                                       PAPØØØ47
C     INPUT OF START NODE JØ                                            PAPØØØ48
C                                                                       PAPØØØ49
    5 READ(IN,1Ø1) JØ                                                   PAPØØØ5Ø
      IF (JØ.LE.Ø) STOP                                                 PAPØØØ51
C                                                                       PAPØØØ52
      CALL SHPTHL(FLIST,DFLIST,KF,N,MJ,JØ,WJ)                           PAPØØØ53
C                                                                       PAPØØØ54
      WRITE(OUTP,2ØØ) JØ                                                PAPØØØ55
      IF (N.GT.Ø) WRITE(OUTP,2Ø1) (I,MJ(I),I=1,N)                       PAPØØØ56
C                                                                       PAPØØØ57
C     INPUT OF END NODE KE                                              PAPØØØ58
C                                                                       PAPØØØ59
    6 READ(IN,1Ø1) KE                                                   PAPØØØ6Ø
      IF (KE.LE.Ø) GO TO 5                                              PAPØØØ61
C                                                                       PAPØØØ62
      CALL SHPATH(JØ,KE,WJ,H,NI,W)                                      PAPØØØ63
C                                                                       PAPØØØ64
      WRITE(OUTP,2Ø2) JØ,KE,MJ(KE)                                      PAPØØØ65
      WRITE(OUTP,2Ø3) (W(I),I=1,NI)                                     PAPØØØ66
      GO TO 6                                                           PAPØØØ67
C                                                                       PAPØØØ68
1ØØ   FORMAT(1ØI8)                                                     PAPØØØ69
1Ø1   FORMAT(I4)                                                       PAPØØØ7Ø
2ØØ   FORMAT(21H1DISTANCES FROM NODE ,I5/)                             PAPØØØ71
2Ø1   FORMAT(1Ø(I7,1H-,I4))                                            PAPØØØ72
2Ø2   FORMAT(//15H PATH FROM NODE,I7,9H TO NODE ,I7,13H (PATHLENGTH=,  PAPØØØ73
     +       I7,1H))                                                    PAPØØ74
2Ø3   FORMAT(1H ,2ØI6)                                                 PAPØØØ75
      END                                                               PAPØØØ76


      SUBROUTINE SHPTHL(FLIST,DFLIST,KF,N,MJ,JØ,WJ)                     SPLØØØØ1
      INTEGER FLIST(2ØØ),DFLIST(2ØØ),KF(51),MJ(5Ø),NJ(5Ø),WJ(5Ø)        SPLØØØØ2
      DATA INF /1ØØØØØØ/                                                SPLØØØØ3
C                                                                       SPLØØØØ4
C     SHPTHL CALCULATES THE SHORTEST PATH LENGTHS (MJ) FROM A SPECIFIC  SPLØØØØ5
C     NODE (JØ) TO ALL OTHER (N-1) NODES IN A NETWORK (FLIST,DFLIST,KF).SPLØØØØ6
C     PREDECESSOR NODES ARE STORED IN WJ.                               SPLØØØØ7
C                                                                       SPLØØØØ8
C     FLIST : FORWARD INDEX LIST                                        SPLØØØØ9
C     DFLIST: DISTANCE LIST                                             SPLØØØ1Ø
```

```
C     KF    : POINTER LIST FOR FLIST AND DFLIST                          SPLØØØ11
C     N     : NUMBER OF NODES                                            SPLØØØ12
C     MJ    : ARRAY OF SHORTEST PATH LENGTHS                             SPLØØØ13
C     JO    : INITIAL NODE, FIRST NODE OF SHORTEST PATH                  SPLØØØ14
C     WJ    : ARRAY OF PREDECESSORS FOR SHORTEST PATH CONSTRUCTION       SPLØØØ15
C     NJ    : DOUBLE ENDED QUEUE FOR NODE DISCUSSION                     SPLØØØ16
C     INF   : A LARGE NUMBER                                             SPLØØØ17
C                                                                        SPLØØØ18
C     INITIAL VALUES OF                                                  SPLØØØ19
C     FLIST,DFLIST,KF: NETWORK FROM INPUT AND MAIN PROGRAM               SPLØØØ2Ø
C     N              : NUMBER OF NODES FROM INPUT AND MAIN PROGRAM       SPLØØØ21
C     JØ             : START NODE FROM INPUT AND MAIN PROGRAM            SPLØØØ22
C                                                                        SPLØØØ23
      DO 1 I=1,N                                                         SPLØØØ24
      MJ(I)=INF                                                          SPLØØØ25
    1 NJ(I)=Ø                                                            SPLØØØ26
      MJ(JØ)=Ø                                                           SPLØØØ27
C                                                                        SPLØØØ28
C     I     : INDEX FOR NODE DISCUSSION, NODE UNDER DISCUSSION           SPLØØØ29
C     NT    : POINTER TO THE END OF DEQUE NJ                             SPLØØØ3Ø
C     MJI   : LOCAL VARIABLE OF MJ(I)                                    SPLØØØ31
C     KFI   : LOCAL VARIABLE OF KF(I)                                    SPLØØØ32
C     KFI1  : LOCAL VARIABLE OF KF(I)+1                                  SPLØØØ33
C     IR    : INDEX FOR ARRAY DISCUSSION                                 SPLØØØ34
C     K     : SUCCESSOR OF NODE I                                        SPLØØØ35
C     MJK   : LOCAL VARIABLE OF MJ(K)                                    SPLØØØ36
C     NJI   : LOCAL VARIABLE OF NJ(I), THE NEXT NODE OF NJ TO BE TAKEN   SPLØØØ37
C             UNDER DISCUSSION                                           SPLØØØ38
C                                                                        SPLØØØ39
      NJ(JØ)=INF                                                         SPLØØØ4Ø
      I=JØ                                                               SPLØØØ41
      NT=JØ                                                              SPLØØØ42
C                                                                        SPLØØØ43
C     OUTER LOOP                                                         SPLØØØ44
C     DISCUSSION OF NODES I                                              SPLØØØ45
C                                                                        SPLØØØ46
    2 KFI=KF(I+1)                                                        SPLØØØ47
      MJI=MJ(I)                                                          SPLØØØ48
      KFI1=KF(I)+1                                                       SPLØØØ49
C                                                                        SPLØØØ5Ø
C     INNER  LOOP                                                        SPLØØØ51
C     DISCUSSION OF SUCCESSORS K                                         SPLØØØ52
C                                                                        SPLØØØ53
      IF (KFI1.GT.KFI) GO TO 6                                           SPLØØØ54
      DO 5 IR=KFI1,KFI                                                   SPLØØØ55
      K=FLIST(IR)                                                        SPLØØØ56
      MJK=MJI+DFLIST(IR)                                                 SPLØØØ57
C     NO DECREASE OF SHORTEST DISTANCES                                  SPLØØØ58
      IF (MJK.GE.MJ(K)) GO TO 5                                          SPLØØØ59
C     DECREASE OF SHORTEST DISTANCES                                     SPLØØØ6Ø
      MJ(K)=MJK                                                          SPLØØØ61
C     PREDECESSOR I OF NODE K                                            SPLØØØ62
      WJ(K)=I                                                            SPLØØØ63
C     NODE K ALREADY IN THE DEQUE NJ ?                                   SPLØØØ64
      IF (NJ(K))  4,3,5                                                  SPLØØØ65
C     NODE K ADDED AT THE END OF THE DEQUE NJ                            SPLØØØ66
    3 NJ(NT)=K                                                           SPLØØØ67
      NT=K                                                               SPLØØØ68
      NJ(K)=INF                                                          SPLØØØ69
      GO TO 5                                                            SPLØØØ7Ø
C     NODE K ADDED AT THE BEGINNING OF THE DEQUE NJ                      SPLØØØ71
    4 NJ(K)=NJ(I)                                                        SPLØØØ72
      NJ(I)=K                                                            SPLØØØ73
    5 CONTINUE                                                           SPLØØØ74
C     NODE I TAKEN FROM THE BEGINNING OF THE DEQUE NJ                    SPLØØØ75
    6 NJI=NJ(I)                                                          SPLØØØ76
      NJ(I)=-NJI                                                         SPLØØØ77
      I=NJI                                                              SPLØØØ78
      IF (I.LT.INF) GOTO 2                                               SPLØØØ79
      RETURN                                                             SPLØØØ8Ø
      END                                                                SPLØØØ81


      SUBROUTINE SHPATH(JØ,KE,WJ,H,NI,W)                                 PATØØØØ1
      INTEGER WJ(5Ø),H(5Ø),W(5Ø)                                         PATØØØØ2
C                                                                        PATØØØØ3
```

```
C     SHPATH CALCULATES THE SHORTEST PATH BETWEEN THE TWO NODES JØ AND  PATØØØØ4
C     KE. SHPATH USES THE INFORMATION IN WJ, THE LIST OF PREDECESSOR    PATØØØØ5
C     NODES. THE NODES OF THE SHORTEST PATH ARE STORED IN W.            PATØØØØ6
C                                                                       PATØØØØ7
C     JØ    : FIRST NODE OF THE SHORTEST PATH                         PATØØØØ8
C     KE    : LAST NODE OF THE SHORTEST PATH                          PATØØØØ9
C     WJ    : ARRAY OF PREDECESSORS FOR SHORTEST PATH CONSTRUCTION     PATØØØ1Ø
C     H     : AUXILIARY ARRAY FOR THE SHORTEST PATH                   PATØØØ11
C     NI    : NUMBER OF NODES OF THE SHORTEST PATH                    PATØØØ12
C     W     : THE SHORTEST PATH                                       PATØØØ13
C                                                                     PATØØØ14
C     I,J   : LOCAL VARIABLE FOR NODE DISCUSSION                      PATØØØ15
C                                                                     PATØØØ16
C     INITIAL VALUES OF                                               PATØØØ17
C     JØ    : FIRST NODE FROM INPUT OR MAIN PROGRAM                   PATØØØ18
C     KE    : LAST NODE FROM INPUT OR MAIN PROGRAM                    PATØØØ19
C     WJ    : FROM SUBROUTINE SHPTHL                                  PATØØØ2Ø
C                                                                     PATØØØ21
      H(1)=KE                                                         PATØØØ22
      I=1                                                             PATØØØ23
      J=KE                                                            PATØØØ24
      IF (JØ.EQ.KE) GO TO 2                                           PATØØØ25
C                                                                     PATØØØ26
    1 I=I+1                                                           PATØØØ27
      J=WJ(J)                                                         PATØØØ28
      H(I)=J                                                          PATØØØ29
      IF (J.NE.JØ) GO TO 1                                            PATØØØ3Ø
C                                                                     PATØØØ31
    2 NI=I                                                            PATØØØ32
C                                                                     PATØØØ33
      DO 3 I=1,NI                                                     PATØØØ34
    3 W(I)=H(NI+1-I)                                                  PATØØØ35
C                                                                     PATØØØ36
      RETURN                                                          PATØØØ37
      END                                                             PATØØØ38
```

ALGORITHM 563
A Program for Linearly Constrained Discrete l_1 Problems

RICHARD H. BARTELS and ANDREW R. CONN
The University of Waterloo

Key Words and Phrases: numerical analysis, overdetermined linear systems, linear constraints, discrete l_1 approximation
CR Categories: 5.13, 5.41
Language: Fortran

DESCRIPTION

The subroutine CL1 given here is a complement of [1], where the theoretical development appears.

REFERENCES

1. BARTELS, R.H., AND CONN, A.R. Linearly constrained discrete l_1 problems. *ACM Trans. Math Softw. 6*, 4 (Dec. 1980), 594-608.

ALGORITHM

[Summary information and a part of the listing are printed here. The complete listing is available from the ACM Algorithms Distribution Service.]

NAME(n): indicates a Fortran module with n records
NAME^T(n): indicates "NAME" is included for testing purposes
NAME^D(n): indicates "NAME" contains test data
Contents: MAIN^T(100), DATFL1^T(65), DATFL2^T(104), CL1(344), SETUP(108), NEWPEN(57), UPDATE(52), MONIT(49), FINDP(175), STEP(144), REFINE(48), DELCOL(51), RESID(108), ADDCOL(70), OBJECT(72), GETV(91), DKHEAP(108), UNIF01(88), ZDRCIN(164), ZDRCOU(166), ZDRGIT(151), ZDRGNV(139), ZDRPOC(143), SASUM(69), SAXPY(55), SCOPY(58), SDOT(54), SROTM(109), SROTMG(181), SSCAL(41), DATAO^D(446)

```
      SUBROUTINE CL1(NEQNS,NEQC,NIQC,NVARS,NACT,IFL,MXS,PSW,          CL1Ø269Ø
     *               E,NER,X,F,EL1N,RES,INDX,W)                       CL1Ø27ØØ
C                                                                     CL1Ø271Ø
      INTEGER IFL,INDX(1),MXS,NACT,NEQC,NEQNS,NER,NIQC,NVARS          CL1Ø272Ø
```

Received 27 October 1976; revised 15 April 1980; accepted 20 May 1980

Author's address: Department of Computer Science, The University of Waterloo, Waterloo, Ontario N2L 3G1, Canada.

```
      LOGICAL PSW                                                        CL1Ø273Ø
      REAL E(NER,1),EL1N,F(1),RES(1),W(1),X(1)                           CL1Ø274Ø
C                                                                        CL1Ø275Ø
C                                                                        CL1Ø276Ø
C     *******************************************************            CL1Ø277Ø
C     A PROGRAM FOR THE SOLUTION IN THE   L1  SENSE                      CL1Ø278Ø
C     OF A LINEAR-EQUATION SYSTEM                                        CL1Ø279Ø
C     (WITH OR WITHOUT LINEAR CONSTRAINTS).                              CL1Ø28ØØ
C     RICHARD H. BARTELS AND ANDREW R. CONN.                             CL1Ø281Ø
C     LATEST UPDATE .... 13 APRIL, 198Ø.                                 CL1Ø282Ø
C                                                                        CL1Ø283Ø
C     DEVELOPMENT OF THIS PROGRAM WAS SUPPORTED                          CL1Ø284Ø
C     IN PART BY U. S. NSF GRANT DCR75-Ø7817                             CL1Ø285Ø
C     AND BY FUNDS FROM THE NATIONAL BUREAU OF STANDARDS,                CL1Ø286Ø
C     AND IN PART BY CANADIAN NRC GRANT A8639.                           CL1Ø287Ø
C                                                                        CL1Ø288Ø
C     +++++ PARAMETERS +++++                                             CL1Ø289Ø
C     ---------------------------------------------------------          CL1Ø29ØØ
C                              INPUT                                     CL1Ø291Ø
C     NAME  TYPE  SUBSCRPT     OUTPUT       DESCRIPTION                  CL1Ø292Ø
C                              SCRATCH                                   CL1Ø293Ø
C     ..........................................................         CL1Ø294Ø
C     NEQNS INT.    NONE       IN       NUMBER OF EQUATIONS             CL1Ø295Ø
C                                       (MAY BE ZERO)                    CL1Ø296Ø
C                                                                        CL1Ø297Ø
C     NEQC  INT.    NONE       IN       NUMBER OF EQUALITY              CL1Ø298Ø
C                                       CONSTRAINTS                      CL1Ø299Ø
C                                       (MAY BE ZERO)                    CL1Ø3ØØØ
C                                                                        CL1Ø3Ø1Ø
C     NIQC  INT.    NONE       IN       NUMBER OF INEQUALITY            CL1Ø3Ø2Ø
C                                       CONSTRAINTS                      CL1Ø3Ø3Ø
C                                       (MAY BE ZERO)                    CL1Ø3Ø4Ø
C                                                                        CL1Ø3Ø5Ø
C     NVARS INT.    NONE       IN       NUMBER OF VARIABLES             CL1Ø3Ø6Ø
C                                                                        CL1Ø3Ø7Ø
C     NACT  INT.    NONE       OUT      NUMBER OF ACTIVE                CL1Ø3Ø8Ø
C                                       EQUATIONS/CONSTRAINTS            CL1Ø3Ø9Ø
C                                       AT TERMINATION                   CL1Ø31ØØ
C                                       (IF ANY, THEIR ASSOCIATED        CL1Ø311Ø
C                                        COLUMN POSITIONS IN  E  WILL    CL1Ø312Ø
C                                        BE LISTED IN  INDX(1)           CL1Ø313Ø
C                                        THROUGH  INDX(NACT) )           CL1Ø314Ø
C                                                                        CL1Ø315Ø
C     IFL   INT.    NONE       OUT      TERMINATION CODE                CL1Ø316Ø
C                                       (SEE BELOW)                      CL1Ø317Ø
C                                                                        CL1Ø318Ø
C     MXS   INT.    NONE       IN       MAXIMUM NUMBER OF STEPS         CL1Ø319Ø
C                                       ALLOWED                          CL1Ø32ØØ
C                                                                        CL1Ø321Ø
C     PSW   LOGIC.  NONE       IN       PRINT SWITCH                    CL1Ø322Ø
C                                       (SEE BELOW)                      CL1Ø323Ø
C                                                                        CL1Ø324Ø
C     E     REAL     2         IN       EQUATION/CONSTRAINT MATRIX      CL1Ø325Ø
C                                       THE FIRST  NEQNS  COLUMNS        CL1Ø326Ø
C                                       (SEE NOTE BELOW) SPECIFY         CL1Ø327Ø
C                                       EQUATIONS, THE REMAINING         CL1Ø328Ø
C                                       COLUMNS (IF ANY) SPECIFY         CL1Ø329Ø
C                                       CONSTRAINTS.                     CL1Ø33ØØ
C                                                                        CL1Ø331Ø
C     NER   INT.    NONE       IN       ROW DIMENSION OF E              CL1Ø332Ø
C                                                                        CL1Ø333Ø
C     X     REAL     1         IN       STARTING VALUES FOR THE         CL1Ø334Ø
C                                       UNKNOWNS (USE ZEROS IF NO        CL1Ø335Ø
C                                       GUESS IS AVAILABLE)              CL1Ø336Ø
C                              OUT      TERMINATION VALUES FOR          CL1Ø337Ø
C                                       THE UNKNOWNS                     CL1Ø338Ø
C                                                                        CL1Ø339Ø
C     F     REAL     1         IN       EQUATION/CONSTRAINT             CL1Ø34ØØ
C                                       RIGHT-HAND SIDES                 CL1Ø341Ø
C                                                                        CL1Ø342Ø
C     EL1N  REAL    NONE       OUT      L1 NORM OF EQUATION             CL1Ø343Ø
C                                       RESIDUALS AT TERMINATION         CL1Ø344Ø
C                                                                        CL1Ø345Ø
C     RES   REAL     1         OUT      EQUATION/CONSTRAINT             CL1Ø346Ø
C                                       RESIDUALS AT TERMINATION         CL1Ø347Ø
C                                                                        CL1Ø348Ø
```

```
C     INDX  INT.     1        OUT     INDEX VECTOR USED TO RECORD       CL1Ø349Ø
C                                     THE ORDER IN WHICH THE COLUMNS    CL1Ø35ØØ
C                                     OF  E  ARE BEING PROCESSED        CL1Ø351Ø
C                                                                       CL1Ø352Ø
C     W     REAL     1        SCR.    WORKING STORAGE                   CL1Ø353Ø
C     ------------------------------------------------------------      CL1Ø354Ø
C                                                                       CL1Ø355Ø
C     +++++ PURPOSE +++++                                               CL1Ø356Ø
C     ------------------------------------------------------------      CL1Ø357Ø
C     THIS SUBROUTINE SOLVES THE   NEQNS BY NVARS                       CL1Ø358Ø
C     SYSTEM OF EQUATIONS                                               CL1Ø359Ø
C                                                                       CL1Ø36ØØ
C                      (A-TRANSPOSE) * X   ==   B                       CL1Ø361Ø
C                                                                       CL1Ø362Ø
C     SUBJECT TO THE  NEQC   CONSTRAINTS                                CL1Ø363Ø
C                                                                       CL1Ø364Ø
C                      (G-TRANSPOSE) * X  .EQ.  H                       CL1Ø365Ø
C                                                                       CL1Ø366Ø
C     AND THE  NIQC  INEQUALITY CONSTRAINTS                             CL1Ø367Ø
C                                                                       CL1Ø368Ø
C                      (C-TRANSPOSE) * X  .GE.  D                       CL1Ø369Ø
C                                                                       CL1Ø37ØØ
C     FOR THE UNKNOWNS  X(1),...,X(NVARS).                              CL1Ø371Ø
C                                                                       CL1Ø372Ø
C     THE PROBLEM MUST BE WELL-POSED, NONTRIVIAL                        CL1Ø373Ø
C     AND OVERDETERMINED IN THE SENSE THAT                              CL1Ø374Ø
C                                                                       CL1Ø375Ø
C                         NVARS .GE. 1                                  CL1Ø376Ø
C                         NEQNS .GE. Ø                                  CL1Ø377Ø
C                         NEQC  .GE. Ø                                  CL1Ø378Ø
C                         NIQC  .GE. Ø                                  CL1Ø379Ø
C               NEQNS+NEQC+NIQC  .GE. NVARS.                            CL1Ø38ØØ
C                                                                       CL1Ø381Ø
C     FURTHER, NO COLUMN OF  A, G  OR  C  SHOULD BE ZERO.               CL1Ø382Ø
C     IF THESE CONDITIONS ARE NOT MET, THE PROGRAM                      CL1Ø383Ø
C     WILL TERMINATE WITHOUT PERFORMING ANY SUBSTANTIVE                 CL1Ø384Ø
C     COMPUTATIONS.                                                     CL1Ø385Ø
C                                                                       CL1Ø386Ø
C     A POINT  X  IS A SOLUTION IF IT MINIMIZES THE EQUATION            CL1Ø387Ø
C     RESIDUALS FROM AMONG ALL POINTS WHICH SATISFY THE                 CL1Ø388Ø
C     CONSTRAINTS.  AT ANY (NONDEGENERATE) SOLUTION                     CL1Ø389Ø
C     THERE WILL BE  NACT  EQUATIONS AND CONSTRAINTS                    CL1Ø39ØØ
C     WHOSE RESIDUALS                                                   CL1Ø391Ø
C                                                                       CL1Ø392Ø
C          (A(I)-TRANSPOSE) * X - B(I)                                  CL1Ø393Ø
C                                                                       CL1Ø394Ø
C          (G(I)-TRANSPOSE) * X - H(I)                                  CL1Ø395Ø
C                                                                       CL1Ø396Ø
C     AND                                                               CL1Ø397Ø
C                                                                       CL1Ø398Ø
C          (C(I)-TRANSPOSE) * X - D(I)                                  CL1Ø399Ø
C                                                                       CL1Ø4ØØØ
C     ARE ZERO.                                                         CL1Ø4Ø1Ø
C                                                                       CL1Ø4Ø2Ø
C     THE COLUMNS OF  (A,G,C)  CORRESPONDING TO THE ZERO RESIDUALS      CL1Ø4Ø3Ø
C     ARE REFERRED TO AS  ACTIVE COLUMNS  THROUGHOUT THIS LISTING.      CL1Ø4Ø4Ø
C     THE NUMBERS OF THE ACTIVE COLUMNS ARE MAINTAINED AS THE           CL1Ø4Ø5Ø
C     ENTRIES  1,...,NACT  OF THE ARRAY  INDX.                          CL1Ø4Ø6Ø
C                                                                       CL1Ø4Ø7Ø
C     A SOLUTION  X  IS FOUND BY MINIMIZING A PIECEWISE                 CL1Ø4Ø8Ø
C     LINEAR PENALTY FUNCTION FORMED FROM THE  L1                       CL1Ø4Ø9Ø
C     NORM OF THE EQUATION RESIDUALS AND THE SUM OF THE                 CL1Ø41ØØ
C     INFEASIBILITIES IN THE CONSTRAINTS.                               CL1Ø411Ø
C     THE MINIMIZATION PROCEEDS IN A STEP-BY-STEP                       CL1Ø412Ø
C     FASHION, TERMINATING AFTER A FINITE NUMBER OF STEPS.              CL1Ø413Ø
C                                                                       CL1Ø414Ø
C     NOTE THAT  A, G  AND  C  APPEAR TRANSPOSED IN THE                 CL1Ø415Ø
C     PROBLEM FORMULATION.  HENCE IT IS THE COLUMNS OF  (A,G,C)         CL1Ø416Ø
C     WHICH DEFINE THE EQUATIONS AND CONSTRAINTS RESPECTIVELY.          CL1Ø417Ø
C                                                                       CL1Ø418Ø
C     THE ARRAY  E  IS A COMPOSITE OF   A, G AND C                      CL1Ø419Ø
C     AND  F  IS A COMPOSITE OF  B, H  AND  D.                          CL1Ø42ØØ
C     E  SHOULD CONTAIN  A  AS ITS FIRST  NEQNS  COLUMNS.               CL1Ø421Ø
C     IT SHOULD CONTAIN  G  AS ITS NEXT  NEQC  COLUMNS AND              CL1Ø422Ø
C     CONTAIN  C  AS ITS REMAINING  NIQC  COLUMNS.                      CL1Ø423Ø
C     SIMILARLY  F  SHOULD CONTAIN  B  AS ITS FIRST                     CL1Ø424Ø
```

```
C     NEQNS  COMPONENTS,  H  AS ITS NEXT  NEQC  COMPONENTS              CL1Ø425Ø
C     AND  D  AS ITS LAST  NIQC  COMPONENTS.                            CL1Ø426Ø
C     ------------------------------------------------------            CL1Ø427Ø
C                                                                       CL1Ø428Ø
C     +++++ ARRAYS +++++                                                CL1Ø429Ø
C     ------------------------------------------------------            CL1Ø43ØØ
C     E  IS TO BE DIMENSIONED AT LEAST    N  BY  M,                     CL1Ø431Ø
C     X                       AT LEAST    N,                            CL1Ø432Ø
C     F                       AT LEAST    M,                            CL1Ø433Ø
C     RES                     AT LEAST    M,                            CL1Ø434Ø
C     INDX                    AT LEAST    M,                            CL1Ø435Ø
C     W                       AT LEAST    ((3*N*N+11*N+2)/2) + (2*M).   CL1Ø436Ø
C                                                                       CL1Ø437Ø
C                                         WHERE  N = NVARS  AND         CL1Ø438Ø
C                                         M = NEQNS+NEQC+NIQC           CL1Ø439Ø
C     ------------------------------------------------------            CL1Ø44ØØ
C                                                                       CL1Ø441Ø
C     +++++ INITIALIZATION +++++                                        CL1Ø442Ø
C     ------------------------------------------------------            CL1Ø443Ø
C     THE USER MUST INITIALIZE                                          CL1Ø444Ø
C                                                                       CL1Ø445Ø
C          NEQNS,NEQC,NIQC,NVARS,MXS,PSW,E,NER,X,F .                    CL1Ø446Ø
C                                                                       CL1Ø447Ø
C     THE FOLLOWING ARE SET BY  CL1                                     CL1Ø448Ø
C     AND DO NOT REQUIRE INITIALIZATION                                 CL1Ø449Ø
C                                                                       CL1Ø45ØØ
C          NACT,INDX,RES .                                              CL1Ø451Ø
C                                                                       CL1Ø452Ø
C     THE ARRAY  W  IS USED AS SCRATCH SPACE.                           CL1Ø453Ø
C     ------------------------------------------------------            CL1Ø454Ø
C                                                                       CL1Ø455Ø
C     +++++ TERMINATION CODES AND INTERMEDIATE PRINTING +++++           CL1Ø456Ø
C     ------------------------------------------------------            CL1Ø457Ø
C     MXS  SETS A LIMIT ON THE NUMBER OF MINIMIZATION STEPS TO BE       CL1Ø458Ø
C     TAKEN.                                                            CL1Ø459Ø
C                                                                       CL1Ø46ØØ
C     UPON TERMINATION  IFL  WILL BE SET ACCORDING TO                   CL1Ø461Ø
C     THE FOLLOWING CODE ...                                            CL1Ø462Ø
C                                                                       CL1Ø463Ø
C             IFL = 1 .... SUCCESSFUL TERMINATION.                      CL1Ø464Ø
C                                                                       CL1Ø465Ø
C             IFL = 2 .... UNSUCCESSFUL TERMINATION.                    CL1Ø466Ø
C                          CONSTRAINTS CANNOT BE SATISFIED.             CL1Ø467Ø
C                          PROBLEM IS INFEASIBLE.                       CL1Ø468Ø
C                                                                       CL1Ø469Ø
C             IFL = 3 .... LIMIT IMPOSED BY  MXS  REACHED               CL1Ø47ØØ
C                          WITHOUT FINDING A SOLUTION.                  CL1Ø471Ø
C                                                                       CL1Ø472Ø
C             IFL = 4 .... PROGRAM ABORTED.                             CL1Ø473Ø
C                          NUMERICAL DIFFICULTIES                       CL1Ø474Ø
C                          DUE TO ILL-CONDITIONING.                     CL1Ø475Ø
C                                                                       CL1Ø476Ø
C             IFL = 5 .... NEQNS, NVARS, NEQC AND/OR                    CL1Ø477Ø
C                          NIQC  HAVE IMPROPER VALUES                   CL1Ø478Ø
C                          OR  E  CONTAINS A ZERO COLUMN.               CL1Ø479Ø
C                                                                       CL1Ø48ØØ
C     IN ALL CASES THE OUTPUT PARAMETERS  X,EL1N AND RES                CL1Ø481Ø
C     WILL CONTAIN THE VALUES WHICH THEY REACHED AT TERMINATION.        CL1Ø482Ø
C                                                                       CL1Ø483Ø
C     INTERMEDIATE PRINTING WILL BE TURNED OFF IF  PSW = .FALSE.        CL1Ø484Ø
C     ON THE OTHER HAND,  DETAILS OF EACH MINIMIZATION CYCLE            CL1Ø485Ø
C     WILL BE PRINTED IF  PSW  IS SET TO  .TRUE.                        CL1Ø486Ø
C     ------------------------------------------------------            CL1Ø487Ø
C                                                                       CL1Ø488Ø
C     +++++ REMARKS AND USER CAUTIONS +++++                             CL1Ø489Ø
C     ------------------------------------------------------            CL1Ø49ØØ
C       1.  BEYOND SOME PRECAUTIONARY STEPS TAKEN IN                    CL1Ø491Ø
C           CERTAIN DIVISIONS, NO SPECIAL                               CL1Ø492Ø
C           OVERFLOW/UNDERFLOW PROTECTION IS PROVIDED.                  CL1Ø493Ø
C       2.  ALL TOLERANCES FOR CHECKING ZEROS AND LINEAR                CL1Ø494Ø
C           DEPENDENCIES ARE DETERMINED FROM THE QUANTITY               CL1Ø495Ø
C           EPS  WHICH APPEARS IN DATA DECLARATIONS IN  CL1             CL1Ø496Ø
C           AND SEVERAL OF ITS SUBROUTINES.   EPS  CAN BE SET TO THE    CL1Ø497Ø
C           LEAST POSITIVE NUMBER SATISFYING  (1.Ø + EPS) .GT. 1.Ø      CL1Ø498Ø
C           IN THE PRECISION OF ARITHMETIC BEING USED.  WITH THIS       CL1Ø499Ø
C           SETTING,  CL1  USES AN EXTREMELY STRICT ZERO TOLERANCE.     CL1Ø5ØØØ
```

```
C           FOR A MORE FORGIVING VERSION OF  CL1,  EPS MAY BE REMOVED   CL1Ø5Ø1Ø
C           FROM THE DATA DEFINITIONS AND INCLUDED AS A USER-SPECIFIED  CL1Ø5Ø2Ø
C           ZERO-TESTING PARAMETER IN THE ARGUMENT LIST.  IN SUCH AN    CL1Ø5Ø3Ø
C           EVENT, IF THE PROBLEM DATA IS GIVEN TO  NDIG  SIGNIFICANT   CL1Ø5Ø4Ø
C           DIGITS, THEN  1Ø.Ø**(-NDIG)  IS A REASONABLE CHOICE         CL1Ø5Ø5Ø
C           FOR THE VALUE OF  EPS.                                      CL1Ø5Ø6Ø
C           OVERFLOW CHECKING PRIOR TO DIVISION IS                      CL1Ø5Ø7Ø
C           DONE USING THE QUANTITY  BIG,  ALSO SPECIFIED AS DATA.      CL1Ø5Ø8Ø
C           BIG  SHOULD BE THE LARGEST REPRESENTABLE FLOATING           CL1Ø5Ø9Ø
C           POINT NUMBER.                                               CL1Ø51Ø0
C       3.  THIS IS A SINGLE PRECISION VERSION OF  CL1.                 CL1Ø511Ø
C           TO CHANGE THIS CODE INTO DOUBLE PRECISION ...               CL1Ø512Ø
C           A.  CHANGE ALL OCCURRENCES OF  - REAL -                     CL1Ø513Ø
C               DECLARATIONS TO  - DOUBLE PRECISION -                   CL1Ø514Ø
C           B.  CHANGE ALL OCCURRENCES OF  - SYSTEM ROUTINES -          CL1Ø515Ø
C               (AS LISTED IN THE HEADING OF EACH SUBROUTINE)           CL1Ø516Ø
C               TO THEIR CORRESPONDING DOUBLE PRECISION VERSIONS        CL1Ø517Ø
C           C.  CHANGE ALL OCCURRENCES OF THE STRINGS  E+  AND  E-      CL1Ø518Ø
C               TO  D+  AND  D-  RESPECTIVELY                           CL1Ø519Ø
C           D.  CHANGE ALL BASIC LINEAR ALGEBRA ROUTINES  (BLAS)        CL1Ø52Ø0
C               TO THEIR DOUBLE-PRECISION EQUIVALENTS                   CL1Ø521Ø
C           E.  BOTH  EPS  AND  BIG  WILL HAVE TO BE CHANGED            CL1Ø522Ø
C               TO THEIR DOUBLE PRECISION EQUIVALENTS                   CL1Ø523Ø
C           F.  THE REFERENCES TO  - IFIX -  IN SUBROUTINES             CL1Ø524Ø
C               - RESID -  AND  - GETV -  MUST BE CHANGED               CL1Ø525Ø
C               FROM THE FORM                                           CL1Ø526Ø
C                    IFIX(FLOAT(K)*UNIF(...))                           CL1Ø527Ø
C               TO THE FORM                                             CL1Ø528Ø
C                    IFIX(FLOAT(K)*SNGL(UNIF(...)))                     CL1Ø529Ø
C           G.  THE REFERENCE TO  - FLOAT -  IN SUBROUTINE              CL1Ø53Ø0
C               - RESID -  MUST BE CHANGED FROM THE FORM                CL1Ø531Ø
C                    FLOAT(...)                                         CL1Ø532Ø
C               TO THE FORM                                             CL1Ø533Ø
C                    DBLE(FLOAT(...))                                   CL1Ø534Ø
C           H.  REMOVE THESE COMMENT CARDS  (3., 3.A.-3.H.).            CL1Ø535Ø
C     ------------------------------------------------------------      CL1Ø536Ø
```

Editor's Note

Owing to the increasing length of algorithms accepted for publication in ACM Transactions, it has become impractical to supply lengths of individual procedures. Instead, lengths of entire algorithms are given on the Algorithms order form.

ALGORITHM 564
A Test Problem Generator for Discrete Linear L_1 Approximation Problems

K.L. HOFFMAN and D.R. SHIER
National Bureau of Standards

Key Words and Phrases: L_1 approximation, least absolute deviation, problem generator, test data
CR Categories: 5.13, 5.41, 5.5
Language: Fortran

DESCRIPTION

The algorithm given here is a complement to [1], where its theoretical development and implementation are described.

REFERENCES

1. HOFFMAN, K.L., AND SHIER, D.R. A test problem generator for discrete linear L_1 approximation problems. *ACM Trans. Math. Softw. 6*, 4 (Dec. 1980), 587–593.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service.]

```
C DESCRIPTION:
C
C    THIS SUBROUTINE GENERATES DATA SETS WHICH CAN BE
C    USED FOR TESTING L1-APPROXIMATION (LEAST ABSOLUTE
C    DEVIATION CURVE-FITTING) COMPUTER CODES.  THE USER
C    CAN SPECIFY
C
C      * PROBLEM SIZE
C      * SOLUTION VECTOR
C      * STATISTICAL DISTRIBUTION FOR COLUMN ELEMENTS
C      * ROW REPETITIONS
C      * DEGENERACY
C      * RANK LOSS
C      * STATISTICAL DISTRIBUTION FOR RESIDUALS
C
```

Received 11 July 1978; revised 20 August 1979; accepted 11 December 1979.
Authors' address: Center for Applied Mathematics, National Bureau of Standards, Washington, DC 20234.

```
C    ON OUTPUT, X = XOPT IS AN OPTIMAL SOLUTION TO THE
C    GENERATED L1-APPROXIMATION PROBLEM:
C
C
C        MINIMIZE  // X*A - RHS // ,
C
C
C    WHERE //...// IS THE L1-NORM AND THE TRANSPOSE OF
C    A REPRESENTS THE DESIGN MATRIX.  FURTHERMORE,
C    X = XOPT IS GUARANTEED TO BE THE UNIQUE OPTIMAL
C    SOLUTION IF RLOSS=Ø.
C
C INPUT ARGUMENTS:
C
C    NOBS -- NUMBER OF OBSERVATIONS
C    NPAR -- NUMBER OF PARAMETERS
C    NXGEN -- INDICATES WHETHER SOLUTION X IS SPECIFIED
C             AS INPUT OR GENERATED
C                = Ø  IF SPECIFIED AS INPUT
C                = 1  IF RANDOMLY GENERATED FROM NORMAL
C                = 2  IF RANDOMLY GENERATED FROM UNIFORM
C    XOPT -- REAL ARRAY OF DIMENSION NPAR - RLOSS WHICH
C            SPECIFIES THE OPTIMAL SOLUTION IF NXGEN IS
C            SET EQUAL TO ZERO
C    XMEAN -- MEAN OF NORMAL DISTRIBUTION IF NXGEN=1,
C             LOWER LIMIT OF UNIFORM DISTRIBUTION IF
C             NXGEN=2
C    XVAR -- STANDARD DEVIATION OF NORMAL DISTRIBUTION
C            IF NXGEN=1, UPPER LIMIT OF UNIFORM DISTRI-
C            BUTION IF NXGEN=2
C    NDIST -- ARRAY OF DIMENSION NPAR - RLOSS WHICH
C             SPECIFIES THE STATISTICAL DISTRIBUTION
C             FOR EACH COLUMN OF THE DESIGN MATRIX
C               = Ø  IF NORMAL
C               = 1  IF UNIFORM
C               = 2  IF ZERO-ONE
C    CLMEAN -- ARRAY OF DIMENSION NPAR - RLOSS WHICH
C              SPECIFIES FOR EACH COLUMN THE MEAN (IF
C              NDIST=Ø), LOWER LIMIT (IF NDIST=1), OR
C              PROPORTION P OF ZERO ENTRIES (IF NDIST=2)
C    CLMVAR -- ARRAY OF DIMENSION NPAR - RLOSS WHICH
C              SPECIFIES FOR EACH COLUMN THE STANDARD
C              DEVIATION (IF NDIST=Ø), THE UPPER LIMIT
C              (IF NDIST=1); NOT USED IF NDIST=2
C
C      ** NOTE **   A CONSTANT COLUMN OF 1@S CAN BE GENERATED
C                   USING CLMEAN = CLMVAR = 1.Ø AND NDIST = 1,
C                   FOR EXAMPLE.
C
C    NREPS -- NUMBER OF TIMES EACH UNIQUE ROW IS TO
C             APPEAR IN THE GENERATED DESIGN MATRIX
C    NDEG -- NUMBER OF ZERO RESIDUALS OUTSIDE OF THE
C            BASIS
C    RLOSS -- NUMBER OF DEPENDENT COLUMNS ADDED
C    NIND -- VALUE SUCH THAT THE DEPENDENT COLUMNS ARE
C            GENERATED BY SUMMING NIND SUCCESSIVE
C            COLUMNS
C    YDIST -- INTEGER WHICH SPECIFIES THE DISTRIBUTION
C             OF (NONZERO) RESIDUALS ASSOCIATED WITH
C             THE NONACTIVE CONSTRAINTS
C               = Ø  IF NORMAL
C               = 1  IF UNIFORM
C    YMEAN -- MEAN OF RESIDUAL DISTRIBUTION (IF YDIST=Ø),
C             OR LOWER LIMIT (IF YDIST=1)
C    YVAR -- STANDARD DEVIATION OF RESIDUAL DISTRIBUTION
C            (IF YDIST=Ø), OR UPPER LIMIT (IF YDIST=1)
C    ISEED -- THE RANDOM NUMBER SEED USED TO
C             INITIATE THE RANDOM NUMBER GENERATOR
C
C      ** NOTE **  IF NXGEN, NDIST OR YDIST ARE OUTSIDE THE
```

```
C                  INDICATED RANGES, THE REQUIRED DISTRIBUTION
C                  IS DEFAULTED TO THAT OF A NORMAL.
C
C OUTPUT ARGUMENTS:
C
C    XOPT -- OPTIMAL SOLUTION VECTOR OF DIMENSION NPAR
C    A -- GENERATED TRANSPOSE OF THE DESIGN MATRIX, WITH
C         NPAR ROWS AND NOBS COLUMNS
C    RHS -- RIGHT HAND SIDE (OR Y) VECTOR OF DIMENSION
C           NOBS
C    IACT -- VECTOR OF DIMENSION NPAR - RLOSS WHICH
C            CONTAINS INDICES OF ACTIVE CONSTRAINT ROWS
C    SUMRES -- OBJECTIVE FUNCTION VALUE, OR SUM OF
C              ABSOLUTE VALUES OF RESIDUALS
C    ISEED -- THE RANDOM NUMBER SEED AVAILABLE
C             UPON TERMINATION OF GENERATION PROCESS
C    IERR -- ERROR FLAG, UPON RETURN
C                = Ø NORMAL EXECUTION
C                = 1 FATAL ERROR
C
C RESTRICTIONS:
C
C    1 .LE. NOBS .LE. 4ØØ
C    1 .LE. NPAR .LE. 25
C    1 .LE. NREPS .LE. NOBS
C    Ø .LE. NDEG .LE. NOBS-NPAR+RLOSS-2
C    Ø .LE. RLOSS .LE. NPAR-1
C    Ø .LE. NIND .LE. NPAR-RLOSS
C    NOBS+RLOSS-NPAR-NDEG MUST BE EVEN
C    NREPS MUST DIVIDE NOBS
```

ALGORITHM 565 PDETWO/PSETM/GEARB: Solution of Systems of Two-Dimensional Nonlinear Partial Differential Equations [D3]

DAVID K. MELGAARD
Kansas State University
and
RICHARD F. SINCOVEC
Boeing Computer Services Company

Key Words and Phrases: partial differential equations, method of lines, finite differences, ordinary differential equations
CR Categories: 5.17
Language: FORTRAN

DESCRIPTION

The algorithm presented here combines PDETWO, PSETM[1], and a modified version of GEARB[2] to form a complete partial differential equations package for systems of time-dependent nonlinear partial differential equations defined over a two-dimensional rectangular region. The descriptions of PDETWO and PSETM, test results, references, and the use of this algorithm are contained in the authors' paper [2]. The ordinary differential equations package, GEARB, is described by Hindmarsh [1], so we only comment on those modifications that we made to GEARB for this algorithm. The principal modification is the addition of a new subroutine to set pointers for dynamic dimensioning of arrays and to check the validity of some of the user input parameters. The interpolation routine in GEARB was also modified so that it would return correctly interpolated solution values at the user-specified output time for time-dependent boundary conditions.

Machine-readable documentation within the package serves as a complete user's guide. The comments in subroutine PDETWO give the essential details on the use of this package, including descriptions of the user-required subroutines. Additional user information is contained in subroutine DRIVEP, the driver routine for the package.

REFERENCES

1. HINDMARSH, H.C. GEARB: Solution of ordinary differential equations having banded Jacobian. Rep. UCID-30059, Rev. 2, Lawrence Livermore Lab., Livermore, Calif., June 1977.

Received 22 November 1976, 16 April 1979, and 16 May 1979.

This work was supported in part by National Science Foundation Grant MCS 76-07684.
Authors' present addresses: D. K. Melgaard, J & M Systems Consultants Ltd., 2403 San Mateo N.E., Albuquerque, NM 87110; R. F. Sincovec, College of Engineering and Applied Science, University of Colorado at Colorado Springs, Colorado Springs, CO 80907.

2. MELGAARD, D.K., AND SINCOVEC, R.F. General software for two-dimensional nonlinear partial differential equations. *ACM Trans. Math. Softw. 7,* 1 (March 1981), 106–125.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service].

```
      SUBROUTINE PDETWO (NPDE,NX,NY,X,Y,T,U,DUDT,DVS,DVST,DV,DH,
     *                   AH,BH,CH,AV,BV,CV,UX,UY,DXI,DXIR,DXIC,
     *                   XAVG,UDVA,UDHR,UDVB,UDHL,UAVGH,UAVGV)
C
C***********************************************************************
C
C       PDETWO IS AN INTERFACE DESIGNED TO SOLVE A SYSTEM OF SECOND
C  ORDER TIME DEPENDENT PARTIAL DIFFERENTIAL EQUATIONS (PDE*S)
C  DEFINED OVER A RECTANGLE IN CONJUNCTION WITH AN ORDINARY
C  DIFFERENTIAL EQUATION (ODE) INTEGRATOR PACKAGE (PRIMARILY A MODI-
C  FIED VERSION OF GEARB(1)).  IT IS AN EXTENSION OF PDEONE (2,3), AN
C  INTERFACE DEVELOPED FOR THE SOLUTION OF ONE-DIMENSIONAL SYSTEMS OF
C  PDE*S.  THIS EXTENSION CONSISTS OF CHANGING FROM THREE POINT
C  DIFFERENCING TO FIVE POINT DIFFERENCING IN ORDER TO APPROXIMATE
C  THE SPATIAL DERIVATIVES IN THE TWO DIMENSIONAL PDE SYSTEM.  TO
C  SOLVE A PDE SYSTEM USING THIS SOFTWARE INTERFACE IN CONJUNCTION
C  WITH AN ODE INTEGRATOR, THE USER IS REQUIRED ONLY TO PROVIDE
C  THE SPATIAL MESH AND DEFINE THE PROBLEM TO BE EVALUATED.  THE
C  INTERFACE WILL THEN FORM AND EVALUATE A SEMIDISCRETE APPROXI-
C  MATION OF THIS SYSTEM OF PDE*S AND THUS PERMIT ONE TO SOLVE
C  THE ORIGINAL PDE SYSTEM AS A SYSTEM OF TIME DEPENDENT ODE*S
C  USING AN ODE INTEGRATOR.
C
C       TO CREATE A DOUBLE PRECISION VERSION OF PDETWO..
C  CHANGE THE REAL STATEMENT BELOW.
C
C***********************************************************************
C  PROBLEM DEFINITION
C***********************************************************************
C
C       PDETWO IS DESIGNED TO BE USED TO SOLVE A COUPLED SYSTEM OF
C  NPDE PDE*S.  THE L-TH PDE OF THIS SYSTEM IS OF THE FORM ..
C
C     DUDT(L) = F (T, X, Y, U, UX, UY, DUXX, DUYY )
C
C  WHERE
C
C       DUDT(L) REPRESENTS THE FIRST PARTIAL OF THE L-TH COMPONENT
C               OF U WITH RESPECT TO T,
C       T       IS THE CURRENT TIME,
C       X,Y     DEFINE THE POSITION IN THE HORIZONTAL AND VERTICAL
C               DIRECTIONS RESPECTIVELY,
C
C       U    =  (  U(1), ... ,  U(NPDE) ),
C       UX   =  ( UX(1), ... , UX(NPDE) ),
C       UY   =  ( UY(1), ... , UY(NPDE) ),
C
C       DUXX    IS AN NPDE BY NPDE ARRAY SUCH THAT
C                          D
C               DUXX(L,K) = -- (DH(L,K)*UX(K)),
C                          DX
C               THE L-TH ROW OF DUXX CORRESPONDS TO THE L-TH PDE,
C
C       DUYY    IS AN NPDE BY NPDE ARRAY SUCH THAT
C                          D
C               DUYY(L,K) = -- (DV(L,K)*UY(K)),
C                          DY
C               THE L-TH ROW OF DUYY CORRESPONDS TO THE L-TH PDE,
.
C
```

```
C  AND
C
C       U(K)     IS THE VALUE OF THE SOLUTION FOR THE K-TH PDE AT
C                (T, X, Y),
C       UX(K)    IS THE FIRST PARTIAL DERIVATIVE OF U(K) WITH RESPECT
C                TO X,
C       UY(K)    IS THE FIRST PARTIAL DERIVATIVE OF U(K) WITH RESPECT
C                TO Y,
C       DH(L,K) IS THE DIFFUSION COEFFICIENT IN THE HORIZONTAL
C                DIRECTION FOR THE L-TH PDE ASSOCIATED WITH U(K),
C       DV(L,K) IS THE DIFFUSION COEFFICIENT IN THE VERTICAL
C                DIRECTION FOR THE L-TH PDE ASSOCIATED WITH U(K).
C
C       DH(L,K) AND DV(L,K) MAY BE FUNCTIONS OF T, X, Y AND U.
C
C  HORIZONTAL BOUNDARY CONDITIONS
C
C     AH(L)*U(L) + BH(L)*UY(L) = CH(L)
C
C  VERTICAL BOUNDARY CONDITIONS
C
C     AV(L)*U(L) + BV(L)*UX(L) = CV(L)
C
C       AH(L), BH(L) AND CH(L) (AV(L), BV(L) AND CV(L)) ARE AT LEAST
C     PIECEWISE CONTINUOUS FUNCTIONS OF THEIR RESPECTIVE VARIABLES.
C     IF BH(L) .NE. Ø (BV(L) .NE. Ø) THEN AH(L), BH(L) AND CH(L)
C     (AV(L), BV(L) AND CV(L)) MAY BE FUNCTIONS OF T, X, Y AND U
C     BUT OTHERWISE THEY MAY ONLY BE FUNCTIONS OF T, X AND Y. NULL
C     BOUNDARY CONDITIONS ARE NOT ALLOWED SINCE PDETWO WAS NOT
C     DESIGNED TO SOLVE HYPERBOLIC PDE*S. A ZERO DIVIDE WILL OCCUR
C     IF A NULL BOUNDARY CONDITION IS SPECIFIED.
C
C  INITIAL CONDITIONS
C
C       THE INITIAL SOLUTION IS DEFINED FOR EACH U(K) TO BE A
C     KNOWN FUNCTION OF X AND Y FOR THE INITIAL TIME T = TØ.
C     THE INITIAL CONDITIONS NEED NOT BE CONSISTENT WITH THE
C     BOUNDARY CONDITIONS AND MAY CONTAIN DISCONTINUITIES.
C
C*************************************************************************
C  USER SUPPLIED ROUTINES
C*************************************************************************
C
C       THE USER MUST PROVIDE A MAIN PROGRAM AND FIVE SUBROUTINES
C  BNDRYH, BNDRYV, DIFFV, DIFFH AND F.  THESE ROUTINES ARE ALL THAT
C  IS REQUIRED TO DEFINE COMPLETELY THE PROBLEM GIVEN ABOVE.
C
C  1)     THE MAIN PROGRAM DEFINES THE SPATIAL MESH, THE INITIAL
C     CONDITIONS AND THE PARAMETERS FOR THE ODE INTEGRATOR, CALLS
C     THE INTEGRATOR AND PRINTS OR PLOTS THE RESULTS.  THE MAIN
C     PROGRAM SHOULD BE CONSTRUCTED AS FOLLOWS ..
C
C     DIMENSION U(***,*,**),X(*),Y(**)
C     DIMENSION WORK(*****),IWORK(****)
C
C         FOR THE DIMENSIONS ABOVE ENTER THE ACTUAL NUMERICAL
C       VALUES FOR * = NX, ** = NY, *** = NPDE , **** = NODE, AND
C       ***** = NPDE * ( NPDE*(NX*2+3) + 13 + NX ) + NX*4 + 4*NODE
C       + LPW + LU
C       WHERE
C           NODE = NPDE*NX*NY
C           LPW  = 1                          IF MITER = Ø
C                = (3*(NX+1)*NPDE-2)*NODE     IF MITER = 1,2
C                = NODE                       IF MITER = 3
C           LU   = NODE*(MORDER+1)
C        SEE MF AND MORDER BELOW FOR THE DEFINITION OF MITER AND
C        MORDER.
C
C        DEFINE ..
C
C        1) THE NUMBER OF MESH POINTS IN THE X (HORIZONTAL) DIRECTION,
C           NX .GE. 3 AND THE NUMBER OF MESH POINTS IN THE Y (VERTICAL)
C           DIRECTION, NY .GE. 3  ( TO CONSERVE STORAGE ORIENT THE
C           PROBLEM SO THAT NX .LE. NY ),
C        2) THE MESH POINTS (I.E. X(1) .LT. X(2) .LT. ... .LT. X(NX)
```

```
C          AND Y(1) .LT. Y(2) .LT. ... .LT. Y(NY)),
C       3) NPDE, THE NUMBER OF PARTIAL DIFFERENTIAL EQUATIONS,
C       4) INITIAL VALUES FOR ALL OF U,
C       5) MORDER, THE MAXIMUM ORDER OF THE METHOD USED IN THE
C          MODIFIED GEARB.  MORDER MUST BE LESS THAN OR EQUAL TO 12
C          IF METH = 1 OR IF METH = 2, IT MUST BE LESS THAN OR EQUAL
C          TO 5.  SEE MF BELOW FOR THE DEFINITION OF METH.
C       6) THE PARAMETERS FOR THE CALL TO THE INTEGRATOR, AND
C       7) THE FIRST SIX LOCATIONS OF THE ARRAY IWORK AS FOLLOWS.
C          IWORK(1) = NPDE
C          IWORK(2) = NX
C          IWORK(3) = NY
C          IWORK(4) = MORDER
C          IWORK(5) = NRWK
C          IWORK(6) = NRIWK
C          WHERE NRWK AND NRIWK ARE THE LENGTHS OF THE ARRAYS WORK
C          AND IWORK RESPECTIVELY.  THEY SHOULD BE AT LEAST EQUAL
C          TO ***** AND **** AS DEFINED IN THE DESCRIPTION OF THE
C          DIMENSIONS.
C
C         IN DETERMINING THE MESH SPACING, THE USER SHOULD BE AWARE
C       THAT THE DIFFERENCE APPROXIMATIONS AT THE INTERIOR POINTS
C       ARE SECOND ORDER ONLY FOR UNIFORM MESHES.  FOR NONUNIFORM
C       MESHES, ONLY FIRST ORDER APPROXIMATIONS SHOULD BE EXPECTED.
C       IN UNUSUAL SITUATIONS (SEE REMARK IN SECTION 3 OF THE
C       ACCOMPANYING PAPER) INVOLVING COUPLED SYSTEMS OF EQUATIONS,
C       CERTAIN APPROXIMATIONS AT THE BOUNDARY ARE ONLY FIRST ORDER.
C       IN THESE UNUSUAL SITUATIONS THE ERROR CAN BE MINIMIZED BY
C       CHOOSING SMALL MESH SPACINGS NEXT TO THE BOUNDARY.  IN ANY
C       CASE THE USER IS ADVISED TO CHOOSE A MESH WHICH IS LOCALLY
C       NEARLY UNIFORM.
C
C         FOR THE MODIFIED GEARB THE PARAMETERS INCLUDE THE DESIRED
C       OUTPUT TIME (TOUT), THE DESIRED LOCAL ACCURACY (EPS), THE
C       NUMBER OF ODE*S (NODE = NPDE*NX*NY), THE INITIAL TIME (TØ),
C       THE INITIAL TIME STEP SIZE (H),THE TYPE OF CALL BEING MADE
C       (INDEX) AND THE TYPE OF INTEGRATION METHOD DESIRED (MF).
C       MF HAS TWO DECIMAL DIGITS, METH AND MITER (MF = 1Ø*METH +
C       MITER).  METH IS THE BASIC METHOD INDICATOR..
C         METH  = 1 MEANS THE ADAMS METHODS.
C         METH  = 2 MEANS THE BACKWARD DIFFERENTIATION FORMULAS
C                   (BDF), OR STIFF METHODS OF GEAR.
C       MITER IS THE ITERATION METHOD INDICATOR..
C         MITER = Ø MEANS FUNCTIONAL ITERATION (NO PARTIAL
C                   DERIVATIVES NEEDED).
C         MITER = 1 MEANS THE CHORD METHOD WITH AN ANALYTIC
C                   JACOBIAN SUPPLIED IN THE USER DEFINED
C                   ROUTINE PDB.  THIS METHOD IS IN GEARB,
C                   BUT IT SHOULD BE AVOIDED.
C         MITER = 2 MEANS THE CHORD METHOD WITH THE JACO-
C                   BIAN CALCULATED IN PSETM.  THIS IS THE
C                   ONLY METHOD WHERE PSETM IS USEFUL.
C         MITER = 3 MEANS THE CHORD METHOD WITH THE JACO-
C                   BIAN REPLACED BY A DIAGONAL APPROXI-
C                   MATION BASED ON A DIRECTIONAL DERIV-
C                   ATIVE.  THIS METHOD IS IN GEARB, BUT
C                   IS NOT GENERALLY USEFUL IN SOLVING PDE*S.
C       SEE COMMENTS IN DRIVEP FOR ADDITIONAL INFORMATION ON THESE
C       PARAMETERS.  NATURALLY THESE PARAMETERS ARE DEPENDENT ON THE
C       INTEGRATOR BEING USED.  FINALLY THE USER SHOULD CALL THE
C       INTEGRATOR.
C
C     CALL DRIVEP (NODE,TØ,H,U,TOUT,EPS,MF,INDEX,WORK,IWORK,X,Y)
C
C       THE INTEGRATOR WILL RETURN THE SOLUTIONS (U) AT T = TOUT
C       TO THE MAIN PROGRAM TO BE PRINTED OR PLOTTED.  IF A CONTINUA-
C       TION TO ANOTHER TOUT IS DESIRED, SIMPLY RESET TOUT AND CALL
C       DRIVEP AGAIN.
C
C     STOP
C     END
```

```
C
C  2)     THE BOUNDARY SUBROUTINES, BNDRYH AND BNDRYV PROVIDE
C     PDETWO WITH THE COEFFICIENTS FOR THE HORIZONTAL AND
C     VERTICAL BOUNDARY CONDITIONS RESPECTIVELY.  THE CONSTRUCTION
C     OF BNDRYV IS ANALOGOUS TO THE CONSTRUCTION OF BNDRYH.  THE
C     USER SHOULD CONSTRUCT BNDRYH AS FOLLOWS ..
C
C     SUBROUTINE BNDRYH (T,X,Y,U,AH,BH,CH,NPDE)
C     DIMENSION U(NPDE), AH(NPDE), BH(NPDE), CH(NPDE)
C
C        THE INCOMING PARAMETERS X AND Y REPRESENT ANY POINT
C       OF X(J) (J = 1,2,...,NX) AND EITHER Y(1) OR Y(NY) RESPECTIVELY.
C       DEFINE THE FUNCTIONS AH(K), BH(K) AND CH(K) (K = 1,2,...,NPDE)
C       FOR THE LOWER (Y=Y(1)) AND UPPER (Y=Y(NY)) BOUNDARIES. NULL
C       BOUNDARY CONDITIONS ARE NOT ALLOWED AND IF SPECIFIED WILL
C       RESULT IN A ZERO DIVIDE.
C
C         TO INSURE A COMPLETELY ACCURATE DEFINITION OF THE BOUNDARY
C       CONDITIONS AT THE CORNER MESH POINTS, THE USER SHOULD
C       CONSULT THE APPROPRIATE DIFFERENCE APPROXIMATIONS (SEE
C       SECTION 3 OF THE ACCOMPANYING PAPER).
C
C     RETURN
C     END
C
C  3)     DIFFH AND DIFFV DEFINE FOR PDETWO THE HORIZONTAL AND VERTICAL
C     DIFFUSION COEFFICIENTS RESPECTIVELY.  THE CONSTRUCTION OF
C     DIFFV IS ANALOGOUS TO THE CONSTRUCTION OF DIFFH.  THE
C     USER SHOULD CONSTRUCT DIFFH AS FOLLOWS ..
C
C     SUBROUTINE DIFFH (T,X,Y,U,DH,NPDE)
C     DIMENSION U(NPDE),DH(NPDE,NPDE)
C
C        IN THIS ROUTINE DEFINE THE DH(L,K) COEFFICIENTS (L,K =
C       1,2,...,NPDE).  THE INCOMING PARAMTERS X AND Y DENOTE EITHER
C       A BOUNDARY POINT OR A MESH MIDPOINT.
C
C     RETURN
C     END
C
C  4)     THE RIGHT HAND SIDE OF THE PDE IS DEFINED FOR PDETWO IN
C     THE SUBROUTINE F.  THIS ROUTINE IS CONSTRUCTED AS FOLLOWS..
C
C     SUBROUTINE F (T,X,Y,U,UX,UY,DUXX,DUYY,DUDT,NPDE)
C     DIMENSION U(NPDE), UX(NPDE), UY(NPDE), DUXX(NPDE,NPDE),
C    * DUYY(NPDE,NPDE), DUDT(NPDE)
C
C        IN THIS ROUTINE, THE INCOMING VALUES X AND Y REPRESENT
C       THE MESH POINT BEING EVALUATED AND UX, UY, DUXX AND DUYY
C       ARE THE VALUES DENOTED IN THE PROBLEM DEFINITION ABOVE.
C       USING THESE VALUES, DEFINE IN DUDT(L) (L = 1,2,...,NPDE)
C       THE RIGHT HAND SIDE OF THE PDE*S.
C
C     RETURN
C     END
C
C*************************************************************************
C  THE INPUT PARAMETERS
C*************************************************************************
C
C  NPDE         IS THE NUMBER OF PARTIAL DIFFERENTIAL EQUATIONS.
C  NX           IS THE NUMBER OF MESH POINTS IN THE HORIZONTAL
C               DIRECTION.
C  NY           IS THE NUMBER OF MESH POINTS IN THE VERTICAL
C               DIRECTION.
C  X            ARE THE MESH POINTS IN THE HORIZONTAL DIRECTION.
C  Y            ARE THE MESH POINTS IN THE VERTICAL DIRECTION.
C  T            IS THE CURRENT TIME.
C  U            CONTAINS THE CURRENT SOLUTION VALUES FOR ALL THE
C               MESH POINTS.
```

```
C  DVS           SAVES THE VERTICAL DIFFUSION COEFFICIENTS FOR
C                FUTURE EVALUATIONS.
C  DVST          SAVES THE VERTICAL DIFFUSION COEFFICIENTS FOR THE
C                FUTURE EVALUATIONS OF THE TOP ROW OF MESH POINTS.
C  DH            RETURNS FROM DIFFH THE HORIZONTAL DIFFUSION COEFFI-
C                CIENTS AND CONTAINS DUXX, THE APPROXIMATIONS TO THE
C                HORIZONTAL DIVERGENCE TERMS, ON CALLS TO F AND STORES
C                THE PREVIOUS VALUE OF DUXX.
C  DV            RETURNS FROM DIFFV THE VERTICAL DIFFUSION COEFFI-
C                CIENTS AND CONTAINS DUYY, THE APPROXIMATIONS TO THE
C                VERTICAL DIVERGENCE TERMS, ON CALLS TO F.
C  AH, BH, CH    CONTAIN THE HORIZONTAL BOUNDARY COEFFICIENTS
C                PASSED TO PDETWO FROM BNDRYH.
C  AV, BV, CV    CONTAIN THE VERTICAL BOUNDARY COEFFICIENTS
C                PASSED TO PDETWO FROM BNDRYV.
C  UX,UY         STORE THE DIFFERENCE APPROXIMATIONS FOR THE FIRST
C                PARTIAL OF U WITH RESPECT TO X AND Y RESPECTIVELY.
C  DXI,DXIR,
C    DXIC        STORE INFORMATION ABOUT THE HORIZONTAL MESH SPACING.
C  XAVG          STORES INFORMATION ABOUT THE AVERAGE BETWEEN TWO MESH
C                POINTS ON THE HORIZONTAL AXIS.
C  UDVA,UDVB,
C    UDHR,UDHL   STORE INFORMATION FOR APPROXIMATING DUXX AND DUYY.
C  UAVGH,UAVGV   CONTAIN U AVERAGES FOR APPROXIMATING THE DIFFUSION
                 COEFFICIENTS AT THE MESH MID-POINTS.
C
C*********************************************************************
C  THE OUTPUT PARAMETERS
C*********************************************************************
C
C  U             CONTAINS SOLUTIONS UPDATED TO TIME T FOR THE
C                BOUNDARY MESH POINTS DEFINED BY DIRICHLET BOUNDARY
C                CONDITIONS.
C  DUDT          IS THE RIGHT HAND SIDE OF THE SYSTEM OF ODE*S PASSED
C                TO THE INTEGRATOR.  THESE VALUES ARE CALCULATED
C                FROM THE CENTERED DIFFERENCE APPROXIMATIONS OF
C                THE SPATIAL VARIABLES.
C
C*********************************************************************
C  REFERENCES
C*********************************************************************
C
C       1.  A. C. HINDMARSH, GEARB.. SOLUTION OF ORDINARY DIFFERENTIAL
C             EQUATIONS HAVING BANDED JACOBIAN, UCID-3ØØ59 REV. 2,
C             LAWRENCE LIVERMORE LABORATORY, P.O.BOX 8Ø8, LIVERMORE,
C             CA 9455Ø, JUNE 1977.
C
C       2.  R.F. SINCOVEC AND N.K. MADSEN, SOFTWARE FOR NONLINEAR
C             PARTIAL DIFFERENTIAL EQUATIONS, ACM TRANSACTIONS ON
C             MATHEMATICAL SOFTWARE, SEPT. 1975, PP. 232-26Ø.
C
C       3.  R.F. SINCOVEC AND N.K. MADSEN, ALGORITHM 494 PDEONE,
C             SOLUTIONS OF SYSTEMS OF PARTIAL DIFFERENTIAL EQUATIONS,
C             ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE, SEPT.
C             1975, PP. 262-263.
C
C*********************************************************************
      SUBROUTINE PSETM (NPDE,NX,NY,X,Y,U,UMAX,USAVE,DUDTR,DUDT,PW,CON,
     *                  MITER,IER,NEBAND,WORK,IWORK)
C
C*********************************************************************
C
C       PSETM IS INTENDED TO BE USED IN CONJUNCTION WITH THE
C  INTERFACE PDETWO FOR THE SOLUTION OF SECOND ORDER TIME DEP-
C  ENDENT PARTIAL DIFFERENTIAL EQUATIONS (PDE*S) DEFINED OVER A
C  RECTANGULAR REGION.  PSETM IS SPECIFICALLY DESIGNED TO REPLACE
C  THE ROUTINE PSETB USED IN THE ORDINARY DIFFERENTIAL EQUATION
C  INTEGRATOR GEARB (1) IN ORDER TO MINIMIZE THE COMPUTATIONS
C  REQUIRED TO GENERATE THE JACOBIAN MATRIX.
C
```

```
C       PSETM IS CALLED BY STIFFP, A ROUTINE IN THE MODIFIED GEARB,
C  WHEN THE INTEGRATION METHOD (MITER=1 OR 2) REQUIRES A JACOBIAN
C  MATRIX.  WHEN MITER=1, THE JACOBIAN IS DETERMINED BY A USER
C  DEFINED ROUTINE, PDB.  THIS METHOD SHOULD BE AVOIDED SINCE IT RE-
C  QUIRES THE USER TO COMPLETELY UNDERSTAND PDETWO.  IF THIS
C  METHOD IS USED, PSETM OFFERS NO ADVANTAGE OVER PSETB.  IF
C  MITER=2, THE JACOBIAN IS APPROXIMATED BY FINITE DIFFERENCES,
C  USING THE APPROXIMATION  J = (DUDT(U+R) - DUDT(U)) / R,
C  WHERE DUDT(U+R) AND DUDT(U) ARE THE VALUES OF THE RIGHT HAND
C  SIDE OF THE PDE*S EVALUATED AT U+R AND U RESPECTIVELY, AND R
C  IS SOME SMALL NUMBER.  PSETM TAKES ADVANTAGE OF THE STRUCTURE
C  OF THE JACOBIAN MATRIX REQUIRED BY PDETWO TO REDUCE THE COMPU-
C  TATIONS NEEDED TO GENERATE THE MATRIX (REQUIRING ONLY 5 * NPDE
C  CALLS TO PDETWO).  WITH EACH CALL TO PDETWO, PSETM DETERMINES
C
C  THE ENTRIES IN THE JACOBIAN FOR THE MESH POINTS IN THE FOLLOWING
C  PATTERN ..
C
C                  X 0 0 0 0 X 0 0 0 0
C                  0 0 0 X 0 0 0 0 X 0
C                  0 X 0 0 0 0 X 0 0 0
C                  0 0 0 0 X 0 0 0 0 X
C                  0 0 X 0 0 0 0 X 0 0
C                  X 0 0 0 0 X 0 0 0 0
C
C  WHERE X REPRESENTS THE POINTS FOR WHICH THE JACOBIAN IS
C  APPROXIMATED.  MITER=2 IS THE RECOMMENDED METHOD.
C
C       TO CREATE A DOUBLE PRECISION VERSION OF PSETM..
C  CHANGE THE REAL STATEMENT BELOW AND CHANGE THE SINGLE PRECISION
C  FUNCTIONS ABS, SQRT, AND AMAX1.
C
C***********************************************************************
C  THE  PARAMETERS
C***********************************************************************
C
C  NPDE          IS THE NUMBER OF PARTIAL DIFFERENTIAL EQUATIONS.
C  NX            IS THE NUMBER OF MESH POINTS IN THE HORIZONTAL
C                DIRECTION.
C  NY            IS THE NUMBER OF MESH POINTS IN THE VERTICAL
C                DIRECTION.
C  X             ARE THE MESH POINTS IN THE HORIZONTAL DIRECTION.
C  Y             ARE THE MESH POINTS IN THE VERTICAL DIRECTION.
C  U             CONTAINS THE CURRENT SOLUTIONS OF THE PARTIAL
C                DIFFERENTIAL EQUATIONS.
C  UMAX          IS THE MAXIMUM U VALUES, WHICH ARE USED TO SCALE
C                THE VALUE IN R AND FOR ERROR CONTROL IN GEARB.
C  USAVE         IS A TEMPORY STORE FOR U VALUES.
C  DUDTR         IS THE VALUE OF DUDT(U+R) RETURNED FROM PDETWO.
C  DUDT          IS THE VALUE OF DUDT(U), WHICH IS PASSED TO PSETM
C                FROM STIFFP.
C  PW            STORES THE APPROXIMATION FOR THE JACOBIAN.
C  CON           IS THE CONSTANT (-H*EL(1)).
C  MITER         INDICATES THE TYPE OF ITERATION METHOD BEING USED
C                BY THE INTEGRATOR.
C                MITER = Ø MEANS FUNCTIONAL ITERATION.
C                MITER = 1 MEANS THE CHORD METHOD WITH AN ANALYTIC
C                          JACOBIAN SUPPLIED IN THE USER DEFINED
C                          ROUTINE PDB.  THIS METHOD IS IN GEARB,
C                          BUT IT SHOULD BE AVOIDED.
C                MITER = 2 MEANS THE CHORD METHOD WITH THE JACO-
C                          BIAN CALCULATED IN PSETM.  THIS IS THE
C                          ONLY METHOD WHERE PSETM IS USEFUL.
C                MITER = 3 MEANS THE CHORD METHOD WITH THE JACO-
C                          BIAN REPLACED BY A DIAGONAL APPROXI-
C                          MATION BASED ON A DIRECTIONAL DERIV-
C                          ATIVE.  THIS METHOD IS IN GEARB, BUT
C                          IS NOT GENERALLY USEFUL IN SOLVING PDE*S.
C  IER           IS AN ERROR INDICATOR USED IN THE ROUTINE DECBR.
C  NEBAND        = 3 * ML + 1
```

```
C  WORK          PROVIDES WORKING STORAGE FOR DYNAMIC DIMENSIONING
C                OF THE ARRAYS IN PDETWO AND THE MODIFIED GEARB.
C  IWORK         IS THE PIVOT VECTOR USED IN THE LU DECOMPOSITION.
C
C***********************************************************************
C  THE VARIABLES
C***********************************************************************
C
C  R             IS A SMALL INCREMENT USED IN EVALUATING THE
C                FUNCTION NEAR THE CURRENT SOLUTION U ( I.E.
C                DUDT (U+R)).
C  RØ            IS A LOWER BOUND ON THE SIZE OF R.
C  D             IS A SMALL NUMBER USED TO APPROXIMATE THE
C                DERIVATIVE.
C  NXNPDE        = NX * NPDE
C
C  T             IS THE TIME BEING USED FOR THE INTEGRATION.
C  H             IS THE CURRENT TIME STEP SIZE BEING USED IN THE
C                INTEGRATION.
C  UROUND        IS THE UNIT ROUNDOFF OF THE MACHINE.
C  EPSJ          IS SQRT(UROUND).
C  ML            IS THE WIDTH OF THE LOWER (AND UPPER) HALF OF THE
C                BAND OF THE JACOBIAN.  ML = (NX + 1) * NPDE - 1.
C  IW1 - IW27    ARE THE SUBSCRIPT POINTERS USED TO INDICATE THE
C                STORAGE LOCATIONS IN WORK OF ARRAYS IN PDETWO AND
C                THE MODIFIED GEARB.
C
C***********************************************************************
C  THE ROUTINES CALLED BY PSETM
C***********************************************************************
C
C  PDB(NODE,T,U,PW,NODE,ML,ML) IS A USER DEFINED ROUTINE WHICH DEFINES
C                THE JACOBIAN IN PW IF MITER=1.  SINCE THIS METHOD IS
C                NOT RECOMMENDED, A DUMMY ROUTINE IS PROVIDED.
C  DECBR(NEBAND,NODE,ML,ML,PW,IWORK,IER) COMPUTES THE LU DECOMPOSITION
C                OF THE JACOBIAN FOR THE DIRECT SOLUTION OF THE
C                JACOBIAN.
C  PDETWO(NPDE,NX,NY,X,Y,T,U,...) IS THE INTERFACE WHICH DESCRETIZES
C                THE SPATIAL VARIABLES OF THE TWO DIMENSIONAL SYSTEM OF
C                PDE*S, THEREBY EVALUATING THE RIGHT HAND SIDE OF
C                THE SYSTEM OF ODE*S.
C
C***********************************************************************
C  REFERENCES
C***********************************************************************
C
C       1.  A. C. HINDMARSH, GEARB.. SOLUTION OF ORDINARY DIFFERENTIAL
C             EQUATIONS HAVING BANDED JACOBIAN, UCID-3ØØ59 REV. 2,
C             LAWRENCE LIVERMORE LABORATORY, P.O.BOX 8Ø8, LIVERMORE,
C             CA 9455Ø, JUNE 1977.
C
C***********************************************************************
```

ALGORITHM 566
FORTRAN Subroutines for Testing Unconstrained Optimization Software [C5], [E4]

JORGE J. MORÉ, BURTON S. GARBOW, and KENNETH E. HILLSTROM
Argonne National Laboratory

Key Words and Phrases: performance testing, systems of nonlinear equations, nonlinear least squares, unconstrained minimization, optimization software
CR Categories: 4.6, 5.15, 5.41
Language: FORTRAN

DESCRIPTION

This is the FORTRAN package of subroutines described in [1] for testing unconstrained optimization software. The following three problem areas are considered.

(1) Zeros of systems of N nonlinear functions in N variables.
(2) Least squares minimization of M nonlinear functions in N variables.
(3) Unconstrained minimization of an objective function with N variables.

The subroutines that define the test functions and starting points depend on the dimension parameters M and N and on the problem number NPROB. We first describe the subroutines for the test functions.

For systems of nonlinear functions,

VECFCN(N, X, FVEC, NPROB)

returns the function values in the N-vector FVEC, and

VECJAC(N, X, FJAC, LDFJAC, NPROB)

returns the Jacobian matrix in the N by N array FJAC. (The parameter LDFJAC is the leading dimension of the array FJAC as defined in the main program.) In order to prevent gross inefficiencies with solvers that only require one function value at a time,

COMFCN(N, K, X, FCNK, NPROB)

returns the Kth function value in FCNK.

For nonlinear least squares,

SSQFCN(M, N, X, FVEC, NPROB)

returns the function values in the M-vector FVEC, and

SSQJAC(M, N, X, FJAC, LDFJAC, NPROB)

returns the Jacobian matrix in the M by N array FJAC.

Received 19 September 1978, 16 May 1979, and 27 July 1979.
This work was performed under the auspices of the U.S. Department of Energy.
Authors' address: Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439.

For unconstrained minimization,

OBJFCN(N, X, F, NPROB)

returns the objective function value in F, and

GRDFCN(N, X, G, NPROB)

returns the gradient components in the N-vector G.

For each problem area, the starting points are generated by

INITPT(N, X, NPROB, FACTOR)

which returns in X the starting point corresponding to the parameters NPROB and FACTOR. If XS denotes the standard starting point, then X will contain FACTOR*XS, except that if XS is the zero vector and FACTOR is not unity, then all the components of X will be set to FACTOR.

To test a code in any of the three problem areas, the user must provide a driver and interface routine. The driver reads in the data that define the dimensions, the problem number, and FACTOR, calls INITPT, and then calls the code of interest and prints out results. The interface routine provides a link between the code with its particular function routine calling sequences and the subroutines for the test functions.

The package includes example drivers and interface routines for each of the problem areas. Sample data are also provided.

REFERENCES

1. MORÉ, J.J., GARBOW, B.S., AND HILLSTROM, K.E. Testing unconstrained optimization software. *ACM Trans. Math. Soft. 7*, 1 (March 1981), 17–41.

ALGORITHM

[Part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service].

```
C     THIS PROGRAM TESTS CODES FOR THE UNCONSTRAINED OPTIMIZATION OF
C     A NONLINEAR FUNCTION OF N VARIABLES. IT CONSISTS OF A DRIVER
C     AND AN INTERFACE SUBROUTINE FCN. THE DRIVER READS IN DATA,
C     CALLS THE UNCONSTRAINED OPTIMIZER, AND FINALLY PRINTS OUT
C     INFORMATION ON THE PERFORMANCE OF THE OPTIMIZER. THIS IS
C     ONLY A SAMPLE DRIVER, MANY OTHER DRIVERS ARE POSSIBLE. THE
C     INTERFACE SUBROUTINE FCN IS NECESSARY TO TAKE INTO ACCOUNT THE
C     FORMS OF CALLING SEQUENCES USED BY THE FUNCTION SUBROUTINES
C     IN THE VARIOUS UNCONSTRAINED OPTIMIZERS.
C
C     SUBPROGRAMS CALLED
C
C       USER-SUPPLIED ..... ENORM,FCN,SOLVER
C
C       MINPACK-SUPPLIED ... GRDFCN,INITPT,OBJFCN
C
C       FORTRAN-SUPPLIED ... DSQRT
C
C     MINPACK. VERSION OF NOVEMBER 1978.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
      SUBROUTINE FCN(N,X,F,GVEC,IFLAG)
      INTEGER N,IFLAG
      DOUBLE PRECISION F
      DOUBLE PRECISION X(N),GVEC(N)
C     **********
C
C     THE CALLING SEQUENCE OF FCN SHOULD BE IDENTICAL TO THE
C     CALLING SEQUENCE OF THE FUNCTION SUBROUTINE IN THE
C     UNCONSTRAINED OPTIMIZER. FCN SHOULD ONLY CALL THE TESTING
C     FUNCTION AND GRADIENT SUBROUTINES OBJFCN AND GRDFCN WITH
C     THE APPROPRIATE VALUE OF PROBLEM NUMBER (NPROB).
C
```

```
C     SUBPROGRAMS CALLED
C
C       MINPACK-SUPPLIED ... GRDFCN,OBJFCN
C
C     MINPACK. VERSION OF JULY 1978.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
C
C     **********
      INTEGER NPROB,NFEV
      COMMON /REFNUM/ NPROB,NFEV
      CALL OBJFCN(N,X,F,NPROB)
      CALL GRDFCN(N,X,GVEC,NPROB)
      NFEV = NFEV + 1
      RETURN
C
C     LAST CARD OF INTERFACE SUBROUTINE FCN.
C
      END
      SUBROUTINE INITPT(N,X,NPROB,FACTOR)
      INTEGER N,NPROB
      DOUBLE PRECISION FACTOR
      DOUBLE PRECISION X(N)
C     **********
C
C     SUBROUTINE INITPT
C
C     THIS SUBROUTINE SPECIFIES THE STANDARD STARTING POINTS FOR THE
C     FUNCTIONS DEFINED BY SUBROUTINE OBJFCN. THE SUBROUTINE RETURNS
C     IN X A MULTIPLE (FACTOR) OF THE STANDARD STARTING POINT. FOR
C     THE SEVENTH FUNCTION THE STANDARD STARTING POINT IS ZERO, SO IN
C     THIS CASE, IF FACTOR IS NOT UNITY, THEN THE SUBROUTINE RETURNS
C     THE VECTOR  X(J) = FACTOR, J=1,...,N.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE INITPT(N,X,NPROB,FACTOR)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE.
C
C       X IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE STANDARD
C         STARTING POINT FOR PROBLEM NPROB MULTIPLIED BY FACTOR.
C
C       NPROB IS A POSITIVE INTEGER INPUT VARIABLE WHICH DEFINES THE
C         NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18.
C
C       FACTOR IS AN INPUT VARIABLE WHICH SPECIFIES THE MULTIPLE OF
C         THE STANDARD STARTING POINT. IF FACTOR IS UNITY, NO
C         MULTIPLICATION IS PERFORMED.
C
C     MINPACK. VERSION OF JULY 1978.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
      SUBROUTINE OBJFCN(N,X,F,NPROB)
      INTEGER N,NPROB
      DOUBLE PRECISION F
      DOUBLE PRECISION X(N)
C     **********
C
C     SUBROUTINE OBJFCN
C
C     THIS SUBROUTINE DEFINES THE OBJECTIVE FUNCTIONS OF EIGHTEEN
C     NONLINEAR UNCONSTRAINED MINIMIZATION PROBLEMS. THE VALUES
C     OF N FOR FUNCTIONS 1,2,3,4,5,1Ø,11,12,16 AND 17 ARE
C     3,6,3,2,3,2,4,3,2 AND 4, RESPECTIVELY.
C     FOR FUNCTION 7, N MAY BE 2 OR GREATER BUT IS USUALLY 6 OR 9.
C     FOR FUNCTIONS 6,8,9,13,14,15 AND 18 N MAY BE VARIABLE,
C     HOWEVER IT MUST BE EVEN FOR FUNCTION 14, A MULTIPLE OF 4 FOR
C     FUNCTION 15, AND NOT GREATER THAN 5Ø FOR FUNCTION 18.
C
```

```
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE OBJFCN(N,X,F,NPROB)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE.
C
C       X IS AN INPUT ARRAY OF LENGTH N.
C
C       F IS AN OUTPUT VARIABLE WHICH CONTAINS THE VALUE OF
C         THE NPROB OBJECTIVE FUNCTION EVALUATED AT X.
C
C       NPROB IS A POSITIVE INTEGER INPUT VARIABLE WHICH DEFINES THE
C         NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18.
C
C     SUBPROGRAMS CALLED
C
C       FORTRAN-SUPPLIED ... DABS,DATAN,DCOS,DEXP,DLOG,DSIGN,DSIN,
C                            DSQRT
C
C     MINPACK. VERSION OF JULY 1978.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
      SUBROUTINE GRDFCN(N,X,G,NPROB)
      INTEGER N,NPROB
      DOUBLE PRECISION X(N),G(N)
C     **********
C
C     SUBROUTINE GRDFCN
C
C     THIS SUBROUTINE DEFINES THE GRADIENT VECTORS OF EIGHTEEN
C     NONLINEAR UNCONSTRAINED MINIMIZATION PROBLEMS. THE PROBLEM
C     DIMENSIONS ARE AS DESCRIBED IN THE PROLOGUE COMMENTS OF OBJFCN.
C
C     THE SUBROUTINE STATEMENT IS
C
C       SUBROUTINE GRDFCN(N,X,G,NPROB)
C
C     WHERE
C
C       N IS A POSITIVE INTEGER INPUT VARIABLE.
C
C       X IS AN INPUT ARRAY OF LENGTH N.
C
C       G IS AN OUTPUT ARRAY OF LENGTH N WHICH CONTAINS THE COMPONENTS
C         OF THE GRADIENT VECTOR OF THE NPROB OBJECTIVE FUNCTION
C         EVALUATED AT X.
C
C       NPROB IS A POSITIVE INTEGER INPUT VARIABLE WHICH DEFINES THE
C         NUMBER OF THE PROBLEM. NPROB MUST NOT EXCEED 18.
C
C     SUBPROGRAMS CALLED
C
C       FORTRAN-SUPPLIED ... DABS,DATAN,DCOS,DEXP,DLOG,DSIGN,DSIN,
C                            DSQRT
C
C     MINPACK. VERSION OF JULY 1978.
C     BURTON S. GARBOW, KENNETH E. HILLSTROM, JORGE J. MORE
```

ALGORITHM 567
Extended-Range Arithmetic and Normalized Legendre Polynomials [A1], [C1]

D. W. LOZIER
National Bureau of Standards
and
J. M. SMITH
George Mason University and National Bureau of Standards

Key Words and Phrases: angular momentum, extended-range arithmetic, Legendre polynomials, overflow, underflow
CR Categories: 3.17, 4.9, 5.12
Language: FORTRAN

DESCRIPTION

This algorithm consists of two logically distinct parts: (1) a package of six FORTRAN subroutines to facilitate the use of a special form of computer floating-point arithmetic that we call extended-range arithmetic; and (2) a FORTRAN subroutine that computes values of normalized Legendre polynomials according to an algorithm that generates (for some inputs) floating-point numbers that are outside the range of any computer. Our desire to produce a robust FORTRAN subroutine to compute these polynomials stimulated the development of the extended-range software package. This package may prove to be useful for many other computations.

Normalized Legendre polynomials are defined by the formula

$$\bar{P}_\nu^\mu(x) = \left\{\left(\nu + \frac{1}{2}\right)\frac{(\nu-\mu)!}{(\nu+\mu)!}\right\}^{1/2} (1-x^2)^{\mu/2}\frac{d^\mu}{dx^\mu}P_\nu(x),$$

where μ and ν are nonnegative integers, x is a real variable lying in the closed interval $[-1, 1]$, and $P_\nu(x)$ is the ordinary Legendre polynomial. These functions satisfy a three-term recurrence relation in μ that is useful in computing $\bar{P}_\nu^\mu(x)$ for fixed ν and x, and sequences $\mu_1, \mu_1 + 1, \ldots, \mu_2$ of values of μ. For stability reasons, the recurrence is applied in the backward direction, starting at $\mu = \nu + 1$ and $\mu = \nu$ and proceeding through $\nu - 1, \nu - 2, \ldots, \mu_1$. The starting value of $\bar{P}_\nu^\nu(x)$ is determined from a first-order recurrence relation, and $\bar{P}_\nu^{\nu+1}(x) = 0$ for all ν, x. When x is close to ± 1 and ν is moderately large, this method fails because of underflow of early values in the recurrence sequence. Nevertheless, the method is attractive because of its inherent stability and simplicity.

A simple extension of floating-point arithmetic, called extended-range arithmetic, overcomes this difficulty. A real number ξ is represented in three storage locations. Two of these hold a number x in ordinary double-precision form; the third holds a signed integer k. The value of ξ is given by

$$\xi = x \times r^k,$$

Received 30 March 1977; 3 January 1979; and 28 February 1979.
Authors' address: National Bureau of Standards, Washington, DC 20234.

where r is the base (or radix) of double-precision numbers. In this (nonunique) representation x is called the *principal part* and k is called the *auxiliary index.* Addition, subtraction, multiplication, and division of extended-range numbers are, of course, trivial to define. The main challenge was to design subroutines that are portable and, at the same time, as efficient as possible. Addition and conversion of extended-range numbers to a decimal form suitable for printing proved to be the most complicated. The subroutine for addition uses nine cases to preclude underflow and overflow of the principal part of the result. The algorithm used in conversion to decimal form requires triple-precision calculation to achieve double-precision accuracy in the converted value of r^k, because of the possible large size of the auxiliary index k.

ACKNOWLEDGMENT

The authors are pleased to acknowledge the aid of R. N. Freemire of the National Bureau of Standards in testing the software and critically reviewing the FORTRAN coding.

REFERENCES

1. SMITH, J.M., OLVER, F.W.J., AND LOZIER, D.W. Extended-range arithmetic and normalized Legendre polynomials. *ACM Trans. Math. Softw. 7,* 1 (March 1981), 93–105.

ALGORITHM

[Usage of the extended-range software is described in the comments in the initializing subroutine SETUP. These are reproduced below. Similarly, usage of NORMP (the subroutine for computing normalized Legendre polynomials) is described below. The complete listings are available from the ACM Algorithms Distribution Service].

```
C *** SUBROUTINE NORMP ***
      SUBROUTINE NORMP(NU, MU1, MU2, ARG, MODE, PN, IPN, ISIG)
      INTEGER NU, MU1, MU2, MODE, IPN, ISIG
      DOUBLE PRECISION ARG, PN
C
C        SUBROUTINE TO CALCULATE NORMALIZED LEGENDRE POLYNOMIALS
C        OF VARYING ORDER AND OF FIXED ARGUMENT AND DEGREE.
C        THE ORDER MU AND DEGREE NU ARE NONNEGATIVE INTEGERS AND THE
C        ARGUMENT IS REAL. THE ALGORITHM REQUIRES THE USE OF
C        NUMBERS OUTSIDE THE NORMAL MACHINE RANGE. THEREFORE,
C        THIS SUBROUTINE EMPLOYS A SPECIAL ARITHMETIC CALLED
C        EXTENDED-RANGE ARITHMETIC.  (SEE SMITH, J.M., OLVER,
C        F.W.J., AND LOZIER, D.W., EXTENDED-RANGE ARITHMETIC
C        AND NORMALIZED LEGENDRE POLYNOMIALS, ACM TRANSACTIONS
C        ON MATHEMATICAL SOFTWARE, 1981).
C
C        IN EXTENDED-RANGE ARITHMETIC EACH NUMBER IS REPRESENTED
C        AS A PAIR (X,IX) WHICH HAS THE VALUE  X*RADIX**IX, WHERE
C        RADIX IS THE BASE IN WHICH CALCULATIONS ARE PERFORMED.  FOR
C        A FULL DESCRIPTION OF EXTENDED-RANGE ARITHMETIC, SEE THE
C        COMMENTS AT THE BEGINNING OF SUBROUTINE SETUP.
C
C        THE NORMALIZED LEGENDRE POLYNOMIAL IS A MULTIPLE OF THE
C        ASSOCIATED LEGENDRE POLYNOMIAL OF THE FIRST KIND
C        WHERE THE NORMALIZING COEFFICIENT IS CHOSEN SO THAT
C        THE INTEGRAL OF THE SQUARE OF THE FUNCTION FROM -1
C        TO +1 IS EQUAL TO 1 (SEE JAHNKE,E. EMDE,F. AND LOSCH,F.,
C        TABLES OF HIGHER FUNCTIONS, MCGRAW HILL, NEW YORK,
C        196Ø, P. 121).
C
C        THE INPUT VALUES TO NORMP ARE NU, MU1, MU2, ARG, AND MODE.
C        THESE MUST SATISFY:
C          1. NU IS A NON-NEGATIVE INTEGER SPECIFYING THE DEGREE.
C          2. MU1 AND MU2 ARE NON-NEGATIVE INTEGERS, WITH
C             MU1.LE.MU2, SPECIFYING THE RANGE OF ORDERS.
C          3. MODE IS AN INTEGER, RESTRICTED TO 1 OR 2,
C             WHICH INDICATES WHICH OF TWO FORMS OF THE
```

```
C             DOUBLE-PRECISION VARIABLE ARG IS INTENDED:
C              A. IF MODE=1 THEN NORMALIZED LEGENDRE(NU,MU,ARG) IS
C                 COMPUTED FOR ALL MU SUCH THAT MU1 .LE. MU .LE. MU2 ,
C                 IN THIS CASE, -1 .LE. ARG .LE. 1  MUST BE SATISFIED.
C              B. IF MODE=2 THEN NORMALIZED LEGENDRE(NU,MU,COS ARG) IS
C                 COMPUTED FOR ALL MU SUCH THAT MU1 .LE. MU .LE. MU2 ,
C                 IN THIS CASE, -PI .LT. ARG .LT. PI  MUST BE SATISFIED.
C
C        THE OUTPUT OF SUBROUTINE NORMP CONSISTS OF THE TWO
C        ARRAYS PN AND IPN AND THE ERROR ESTIMATE ISIG. THE
C        NORMALIZED LEGENDRE POLYNOMIALS ARE STORED AS EXTENDED-
C        RANGE NUMBERS WITH DOUBLE-PRECISION PRINCIPAL PARTS
C        IN ARRAY PN AND AUXILIARY INDICES IN ARRAY IPN. THUS
C             (PN(1),IPN(1))=NORMALIZED LEGENDRE(NU,MU1,X)
C             (PN(2),IPN(2))=NORMALIZED LEGENDRE(NU,MU1+1,X)
C                .
C                .
C             (PN(K),IPN(K))=NORMALIZED LEGENDRE(NU,MU2,X)
C        WHERE K=MU2-MU1+1 AND X=ARG IF MODE=1, OR X=COS ARG IF MODE=2.
C        FINALLY, ISIG IS AN ESTIMATE OF THE NUMBER OF  DECIMAL DIGITS
C        LOST THROUGH ROUNDING ERRORS IN THE COMPUTATION. IF ARG IS
C        ACCURATE TO N SIGNIFICANT DECIMALS, THEN THE COMPUTED FUNCTION
C        VALUES ARE ACCCURATE TO N-ISIG SIGNIFICANT DECIMALS (EXCEPT IN
C        NEIGHBORHOODS OF ZEROS OF THE FUNCTIONS).
C *** SUBROUTINE SETUP ***
C USAGE
C                    CALL SETUP(IRAD,NRADPL,DZERO,NBITS)
C DESCRIPTION
C
C   SUBROUTINE  SETUP  MUST BE CALLED PRIOR TO CALLING ANY OTHER
C EXTENDED-RANGE SUBROUTINE. IT CALCULATES AND STORES SEVERAL
C MACHINE-DEPENDENT CONSTANTS IN COMMON BLOCKS. THE USER MUST
C SUPPLY FOUR CONSTANTS THAT PERTAIN TO HIS PARTICULAR COMPUTER.
C THE CONSTANTS ARE
C
C          IRAD = THE INTERNAL BASE OF DOUBLE-PRECISION
C                 ARITHMETIC IN THE COMPUTER.
C        NRADPL = THE NUMBER OF RADIX PLACES CARRIED IN
C                 THE DOUBLE-PRECISION REPRESENTATION.
C         DZERO = THE SMALLEST OF 1/DMIN, DMAX, DMAXLN WHERE
C                 DMIN = THE SMALLEST POSITIVE DOUBLE-PRECISION
C                 NUMBER OR AN UPPER BOUND TO THIS NUMBER,
C                 DMAX = THE LARGEST DOUBLE-PRECISION NUMBER
C                 OR A LOWER BOUND TO THIS NUMBER,
C                 DMAXLN = THE LARGEST DOUBLE-PRECISION NUMBER
C                 SUCH THAT DLOG(DMAXLN) CAN BE COMPUTED BY THE
C                 FORTRAN SYSTEM (ON MOST SYSTEMS DMAXLN = DMAX).
C         NBITS = THE NUMBER OF BITS (EXCLUSIVE OF SIGN) IN
C                 AN INTEGER WORD.
C
      SUBROUTINE SETUP(IRAD, NRADPL, DZERO, NBITS)
      INTEGER IRAD, NRADPL, NBITS
      DOUBLE PRECISION DZERO
C
C   THIS IS THE SETTING-UP SUBROUTINE FOR A PACKAGE OF SUBROUTINES
C THAT FACILITATE THE USE OF EXTENDED-RANGE ARITHMETIC. EXTENDED-RANGE
C ARITHMETIC ON A PARTICULAR COMPUTER IS DEFINED ON THE SET OF NUMBERS
C OF THE FORM
C
C               (X,IX) = X*RADIX**IX
C
C WHERE X IS A DOUBLE-PRECISION NUMBER CALLED THE PRINCIPAL PART,
C IX IS AN INTEGER CALLED THE AUXILIARY INDEX, AND RADIX IS THE
C INTERNAL BASE OF THE DOUBLE-PRECISION ARITHMETIC.  OBVIOUSLY,
C EACH REAL NUMBER IS REPRESENTABLE WITHOUT ERROR BY MORE THAN ONE
C EXTENDED-RANGE FORM.  CONVERSIONS BETWEEN  DIFFERENT FORMS ARE
C ESSENTIAL IN CARRYING OUT ARITHMETIC OPERATIONS.  WITH THE CHOICE
C OF RADIX WE HAVE MADE, AND THE SUBROUTINES WE HAVE WRITTEN, THESE
C CONVERSIONS ARE PERFORMED WITHOUT ERROR (AT LEAST ON MOST COMPUTERS).
C (SEE SMITH, J.M., OLVER, F.W.J., AND LOZIER, D.W., EXTENDED-RANGE
C ARITHMETIC AND NORMALIZED LEGENDRE POLYNOMIALS, ACM TRANSACTIONS ON
C MATHEMATICAL SOFTWARE, 1981).
C
C   AN EXTENDED-RANGE NUMBER  (X,IX)  IS SAID TO BE IN ADJUSTED FORM IF
C X AND IX ARE ZERO OR
C
C           RADIX**(-L) .LE. DABS(X) .LT. RADIX**L
C
```

```
C IS SATISFIED, WHERE L IS A COMPUTER-DEPENDENT INTEGER DEFINED IN THIS
C SUBROUTINE. TWO EXTENDED-RANGE NUMBERS IN ADJUSTED FORM CAN BE ADDED,
C SUBTRACTED, MULTIPLIED OR DIVIDED (IF THE DIVISOR IS NONZERO) WITHOUT
C CAUSING OVERFLOW OR UNDERFLOW IN THE PRINCIPAL PART OF THE RESULT.
C WITH PROPER USE OF THE EXTENDED-RANGE SUBROUTINES, THE ONLY OVERFLOW
C THAT CAN OCCUR IS INTEGER OVERFLOW IN THE AUXILIARY INDEX. IF THIS
C IS DETECTED, THE EXTENDED-RANGE SOFTWARE PRINTS A MESSAGE AND STOPS.
C
C   MULTIPLICATION AND DIVISION IS PERFORMED BY SETTING
C
C                  (X,IX)*(Y,IY) = (X*Y,IX+IY)
C OR
C                  (X,IX)/(Y,IY) = (X/Y,IX-IY).
C
C PRE-ADJUSTMENT OF THE OPERANDS IS ESSENTIAL TO AVOID
C OVERFLOW OR  UNDERFLOW OF THE PRINCIPAL PART. SUBROUTINE
C ADJUST (SEE BELOW) MAY BE CALLED TO TRANSFORM ANY EXTENDED-
C RANGE NUMBER INTO ADJUSTED FORM.
C
C   ADDITION AND SUBTRACTION REQUIRE THE USE OF SUBROUTINE ADD
C (SEE BELOW).  THE INPUT OPERANDS NEED NOT BE IN ADJUSTED FORM.
C HOWEVER, THE RESULT OF ADDITION OR SUBTRACTION IS RETURNED
C IN ADJUSTED FORM.  THUS, FOR EXAMPLE, IF (X,IX),(Y,IY),
C (U,IU),  AND (V,IV) ARE IN ADJUSTED FORM, THEN
C
C                  (X,IX)*(Y,IY) + (U,IU)*(V,IV)
C
C CAN BE COMPUTED AND STORED IN ADJUSTED FORM WITH NO EXPLICIT
C CALLS TO ADJUST.
C
C   WHEN AN EXTENDED-RANGE NUMBER IS TO BE PRINTED, IT MUST BE
C CONVERTED TO AN EXTENDED-RANGE FORM WITH DECIMAL RADIX.  SUBROUTINE
C CONVRT IS PROVIDED FOR THIS PURPOSE.
C
C   THE SUBROUTINES CONTAINED IN THIS PACKAGE ARE
C
C *** SUBROUTINE ADD ***
C USAGE
C                   CALL ADD(X,IX,Y,IY,Z,IZ)
C DESCRIPTION
C                   FORMS THE EXTENDED-RANGE SUM  (Z,IZ) =
C                   (X.IX) + (Y,IY).  (Z,IZ) IS ADJUSTED
C                   BEFORE RETURNING. THE INPUT OPERANDS
C                   NEED NOT BE IN ADJUSTED FORM, BUT THEIR
C                   PRINCIPAL PARTS MUST SATISFY
C                   RADIX**(-2L).LE.DABS(X).LE.RADIX**(2L),
C                   RADIX**(-2L).LE.DABS(Y).LE.RADIX**(2L).
C
C *** SUBROUTINE ADJUST ***
C USAGE
C                   CALL ADJUST(X,IX)
C DESCRIPTION
C                   TRANSFORMS (X,IX) SO THAT
C                   RADIX**(-L) .LE. DABS(X) .LT. RADIX**L.
C                   ON MOST COMPUTERS THIS TRANSFORMATION DOES
C                   NOT CHANGE THE MANTISSA OF X PROVIDED RADIX IS
C                   THE NUMBER BASE OF DOUBLE-PRECISION ARITHMETIC.
C
C *** SUBROUTINE CNV21Ø ***
C USAGE
C                   CALL CNV21Ø(K,Z,J)
C DESCRIPTION
C                   GIVEN K THIS SUBROUTINE COMPUTES J AND Z
C                   SUCH THAT  RADIX**K = Z*1Ø**J, WHERE Z IS IN
C                   THE RANGE 1/1Ø .LE. Z .LT. 1.
C                   THE VALUE OF Z WILL BE ACCURATE TO FULL
C                   DOUBLE PRECISION PROVIDED THE NUMBER
C                   OF DECIMAL PLACES IN THE LARGEST
C                   INTEGER PLUS THE NUMBER OF DECIMAL
C                   PLACES CARRIED IN DOUBLE PRECISION DOES NOT
C                   EXCEED 6Ø. CNV21Ø IS CALLED BY SUBROUTINE
C                   CONVRT WHEN NECESSARY. THE USER SHOULD
C                   NEVER NEED TO CALL CNV21Ø DIRECTLY.
C
C *** SUBROUTINE CONVRT ***
C USAGE
C                   CALL CONVRT(X,IX)
```

```
C DESCRIPTION
C                   CONVERTS (X,IX) = X*RADIX**IX
C                   TO DECIMAL FORM IN PREPARATION FOR
C                   PRINTING, SO THAT (X,IX) = X*1Ø**IX
C                   WHERE 1/1Ø .LE. DABS(X) .LT. 1
C                   IS RETURNED, EXCEPT THAT IF
C                   (DABS(X),IX) IS BETWEEN RADIX**(-2L)
C                   AND RADIX**(2L) THEN THE REDUCED
C                   FORM WITH IX = Ø IS RETURNED.
C
C *** SUBROUTINE REDUCE ***
C USAGE
C                   CALL REDUCE(X,IX)
C DESCRIPTION
C                   IF
C                   RADIX**(-2L) .LE. (DABS(X),IX) .LE. RADIX**(2L)
C                   THEN REDUCE TRANSFORMS (X,IX) SO THAT IX=Ø.
C                   IF (X,IX) IS OUTSIDE THE ABOVE RANGE,
C                   THEN REDUCE TAKES NO ACTION.
C                   THIS SUBROUTINE IS USEFUL IF THE
C                   RESULTS OF EXTENDED-RANGE CALCULATIONS
C                   ARE TO BE USED IN SUBSEQUENT ORDINARY
C                   DOUBLE-PRECISION CALCULATIONS.
C
C   LUERR IS THE OUTPUT UNIT NUMBER FOR ERROR MESSAGES, SET
C EQUAL TO 6 BELOW. SETUP DETERMINES THE OTHER COMMON VARIABLES
C FROM THE INPUT.
      COMMON /EXR1/ LUERR
      DOUBLE PRECISION RADIX, RADIXL, RAD2L, DLC1ØR
      COMMON /EXR2/ RADIX, RADIXL, RAD2L, DLG1ØR, L, L2, KMAX
      COMMON /EXR3/ NLG1Ø2, MLG1Ø2, LG1Ø2(21)
```

ALGORITHM 568
PDS—A Portable Directory System

DAVID R. HANSON
The University of Arizona

Key Words and Phrases: file system, UNIX, RATFOR
CR Categories: 4.19, 4.33, 4.35, 4.41
Language: RATFOR (FORTRAN)

1. DESCRIPTION

PDS is a set of procedures that provides a machine-independent method of file specification. PDS provides capabilities beyond those provided by many vendor-supplied systems. In addition, because PDS is portable, additional capabilities, such as protection schemes, file usage statistics, or some of the functions of a source code control system [5], can be added easily.

The basic function of PDS is to maintain a useful directory structure and provide a set of primitives for manipulating that structure. The PDS directory structure is identical to the tree structure of the UNIX [4] file system, and many of the PDS primitives are identical to UNIX primitives. PDS is, in large part, a portable implementation of the UNIX directory system. Besides PDS's machine-independence, the major differences are the extensibility of PDS and, as described in the next section, its i/o independence.

In the simplest terms, PDS provides a directory structure and a *mapping* from machine-independent file names to machine-dependent names. It deals only with the information *describing* a file; it does not use or manipulate actual files in any way. The importance of this approach is that PDS is used to specify a file but does not participate in the actual i/o to that file. Consequently, there is no impact on i/o efficiency when PDS is used.

PDS manipulates a rooted tree structure in which the leaves are files or directories and the nodes are directories. A directory is simply a list of files and directories. An example is shown in Figure 1, in which circles indicate directories and squares indicate files. The root of the tree is denoted by "/", and files and directories are denoted by their "path," which specifies their absolute position in the tree. A path is composed of the names of the nodes on the path from the root to the desired file or directory. The path components are separated by slashes; for example, the path for file "file2" in Figure 1 is "/m/n/file2." The directory entries "." and ".." refer, respectively, to the directory itself and to the immediate ancestor. These names may be used as path components, providing an explicit means of using the structural properties of the tree. If a path does not begin with "/", it is taken to be rooted at the "current directory." For example, with the

Received December 1979; revised September 1980; accepted November 1980

This work was supported by the National Science Foundation under grant MCS78-02545.
Author's address: Department of Computer Science, The University of Arizona, Tucson, AZ 85721.

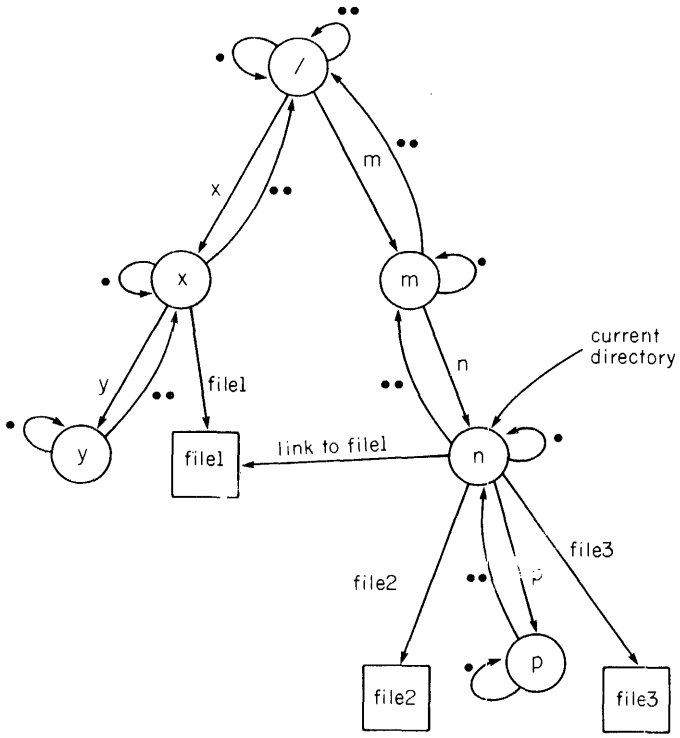


Fig. 1. A directory system.

current directory at "n" in Figure 1, the path "file3" is equivalent to "/m/n/file3."

Files and directories are equivalent with the exception that directories are manipulated by PDS and files are not. Files simply "contain" the machine-dependent names of the actual files. These machine-dependent names are referred to as *host names*. The basic function of PDS is to map paths into host names.

There are ten PDS primitives; they are summarized in Table I. Detailed descriptions of each primitive are given in [1] and in the machine-readable comments. The most important primitives are *openf* and *creatf*, which map a path to a host name and perform machine-dependent operations concerned with preparing files for i/o. The rest are concerned primarily with manipulating the directory structure.

openf obtains the host name corresponding to *path* and calls a machine-dependent routine to actually open the file. *mode* indicates how the file is to be opened (e.g., "read", "write", "append", etc.). *mode* is not modified by *openf*, so it can be whatever is most appropriate on the host system. *openf* returns what the machine-dependent open routine returns, which is whatever is most useful as an argument in calling machine-dependent i/o routines that perform the actual i/o (e.g., **read** and **write**). Typically, a channel number or FORTRAN unit number is returned.

creatf is similar to *openf*, except that *path* is created along with a host name and file. The generation of the host name is performed automatically. After the file is created, it is opened according to *mode* as if *openf* had been called.

openf and *creatf* are insensitive to the values of *mode*. It is, however, useful to introduce standard values representing common file usage, such as reading and writing. This approach promotes machine-independence in programs that use PDS and is used, for example, with the programs described in [3]. Such *mode* values are scrutinized by the machine-dependent i/o interface routines, however, permitting PDS to be used in almost any environment—not just those in which i/o capabilities conform to preconceived notions of "common" file usage.

PDS has *no* other i/o primitives; it is completely independent of the actual i/o. Thus, additional overhead is incurred in opening files, but none is incurred in accessing them. The effect of the additional overhead is minimal since the former

Table I. PDS Primitives

chdir(*path*)	change current directory to *path*
chds(*root, ilist*)	change to another directory system
creatf(*path, mode*)	create *path* and open it for i/o
link(*path*1, *path*2)	make a link to *path*1 named *path*2
mkdir(*path*)	make a directory named *path*
mkds(*root, ilist*)	make a directory system
mkfile(*path, hname*)	make a file *path* with host name *hname*
openf(*path, mode*)	open *path* for i/o according to *mode*
stat(*path, array*)	return information about *path*
unlink(*path*)	unlink *path*

is performed much less frequently than the latter. The contribution of PDS is a portable hierarchical directory system with absolutely no time impact on i/o efficiency.

2. USAGE

PDS is packaged as a set of RATFOR [2, 3] (and hence FORTRAN) functions and subroutines, which is loaded with the program or system that uses it. PDS is useful in programs or systems that use named files extensively, or where a machine-independent means of naming files and a flexible directory structure would enhance utility. Another use would be in a set of tools, such as those described in [3], implemented on computers with limited file systems.

The following program, saveall, illustrates a typical use of PDS. It copies all of the files in the current directory to files of the same name in the directory "../backup". saveall is written in RATFOR in the style of [3].

```
# saveall - save all files in ../backup
  character dline(MAXLINE), bname(MAXLINE)
  integer fd, fdi, fdo
  integer openf, creatf, getlin
  string dot "."
  string backup "../backup/"

  fd = openf(dot, READ)
  while (getlin(dline, fd) ~= EOF) {
    fdi = openf(dline(5), READ)
    call strcat(backup, dline(5), bname)
    fdo = creatf(bname, WRITE)
    call fcopy(fdi, fdo)
    call close(fdi)
    call close(fdo)
    }
  call close(fd)
  end
```

Uppercase names denote defined constants; for example, a typical value for *MAXLINE* might be 80. The **string** statement declares a character array large enough to accommodate the indicated character string. The program begins by opening the current directory for reading using *openf*. The **while** loop reads the directory, line-by-line, until end-of-file, placing each line in the character array *dline*. As described in the next section, each line in a directory contains a file name beginning in column 5. Thus the body of the **while** loop opens each file for reading (*openf*), constructs the path name of the backup copy (*strcat*), creates the backup file and opens it for writing (*creatf*), copies the file (*fcopy*), and finally closes the original and backup files (*close*).

This program makes use of the i/o routines described in [3] (*getlin* and *close*), but any other set of routines—including standard FORTRAN **read** and **write** statements—could be used.

Another example of the use of PDS is as a simple command preprocessor that translates machine-independent system commands to the appropriate host command sequence. In commands, files are referred to by their PDS names and are

mapped to the corresponding host names during the generation of host commands. For example, the command

list *files* . . .

lists the named files on the line printer. The preprocessor translates the **list** into whatever is appropriate on the host system. For example,

list ../backup/saveall.r saveall.r

results in the DEC-10 command sequence

r queue
ilist.120, save.rat

where "ilist.120" and "save.rat" are the host names for "../backup/saveall.r" and "saveall.r", respectively.

3. IMPLEMENTATION

PDS is designed to use only simple sequential i/o (i.e., FORTRAN i/o). It is implemented in RATFOR for portability reasons and because FORTRAN is the only widely available high-level language to which calls from other languages can be made.

The implementation is similar to the implementation of the UNIX file system [7]. The basic technique is to separate the information about a file from the presence of that file in a directory. An *i-node* is associated with each file and directory, and it contains all of the information, including the host name, concerning the file. All of the i-nodes are stored in a single file in which i-node n is line n. This file, referred to as the *i-list*, is stored in character format so that it can be read easily with FORTRAN i/o or its equivalent.

The i-list maps i-node number, or i-number, to a host name. Directories provide a map of PDS name to i-number, thereby completing the mapping from PDS name to host name. A directory is a file of lines, each line containing an i-number and the corresponding PDS name.

Further implementation details are given in [1].

4. INSTALLATION

PDS consists of about 1000 lines of RATFOR including comments. Of this total, however, 300 lines are utility subroutines commonly available in a RATFOR environment. After preprocessing by RATFOR, the resulting FORTRAN code is about 1200 lines in length. The resulting FORTRAN conforms to the portable subset of ANSI standard FORTRAN as defined by PFORT [6].

On a DEC-10 (36-bit words, 512-word pages), the code area for the entire system occupies 6 pages and the data area occupies 17 pages. Small adjustments in the size of the data area can be made by changing various parameters, although the tendency seems to be to *increase* those parameters.

As mentioned in the previous section, PDS operates using sequential i/o only. It is therefore possible to use FORTRAN i/o, although in practice modifications are necessary to avoid having to use FORTRAN unit numbers and to make use of named host files. PDS is written to use the i/o interface described in [3]; a FORTRAN version of these routines is provided along with suggested modifications.

Installation of the system using the FORTRAN version of the i/o interface can be accomplished in 1-2 man-days. The implementation of more sophisticated i/o systems requires substantial time investment, typically 3-6 man-months depending on the target system. This is unnecessary unless heavy use of the RATFOR i/o interface is anticipated. System-specific i/o systems are supplied for the DEC-10 and Cyber 175.

PDS can be modified to use standard FORTRAN i/o facilities directly (i.e., **read** and **write** statements). The modification requires replacing parts of the RATFOR i/o interface—the 300 lines of code mentioned above—with the appro-

priate standard FORTRAN i/o statements. While these modifications have *not* been made, similar experience suggests that 1-2 man-weeks is a conservative estimate of the effort required.

PDS is distributed with the following components:

PDS written in RATFOR
PDS written in FORTRAN (RATFOR output)
RATFOR and RATFOR i/o system written in RATFOR
RATFOR and RATFOR i/o system written in FORTRAN
DEC-10 i/o system
Cyber 175 i/o system

REFERENCES

1. HANSON, D.R. A portable file directory system. *Softw. Pract. Exper. 10,* 8 (Aug. 1980), 623–634.
2. KERNIGHAN, B.W. Ratfor—A preprocessor for a rational Fortran. *Softw. Pract. Exper. 5,* 4 (Dec. 1975), 396–406.
3. KERNIGHAN, B.W., AND PLAUGER, P.J. *Software Tools.* Addison-Wesley, Reading, Mass., 1976.
4. RITCHIE, D.M., AND THOMPSON, K. The UNIX timesharing system. *Commun. ACM 17,* 7 (July 1974), 365–375.
5. ROCHKIND, M.J. The source code control system. *IEEE Trans. Softw. Eng. SE-1,* 4 (Dec. 1975), 364–370.
6. RYDER, B.G. The PFORT verifier. *Softw. Pract. Exper. 4,* 4 (Dec. 1974), 359–377.
7. THOMPSON, K. UNIX implementation. *Bell Syst. Tech. J. 57,* 6 (July 1978), 1931–1946.

ALGORITHM

[The complete algorithm is available from the ACM Algorithms Distribution Service].

ALGORITHM 569
COLSYS: Collocation Software for Boundary-Value ODEs [D2]

U. ASCHER
University of British Columbia, Canada
and
J. CHRISTIANSEN and R. D. RUSSELL
Simon Fraser University, British Columbia, Canada

Key Words and Phrases: ordinary differential equations, boundary-value problems, collocation, B-spline, mesh selection, error estimates, damped Newton's method, general-purpose code
CR Categories: 5.17
Language: FORTRAN

DESCRIPTION

This package is a complement to [1] where its usage is described and demonstrated.

COLSYS uses the following non-American National Standard FORTRAN conventions:

(1) No run-time subscript range checking.
(2) Simple expressions used as subscripts.
(3) Specifications of array values in DATA statements (used in subroutine CONSTS) by referring to the array name only.

REFERENCES

1. ASCHER, U., CHRISTIANSEN, J., AND RUSSELL, R.D. Collocation software for boundary value ODEs. *ACM Trans. Math. Softw.* 7, 2 (June 1981), 209–222

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service .]

```
      SUBROUTINE COLSYS (NCOMP, M, ALEFT, ARIGHT, ZETA, IPAR, LTOL,
     1            TOL, FIXPNT, ISPACE, FSPACE, IFLAG, FSUB,
     2            DFSUB, GSUB, DGSUB, SOLUTN)
C
C
C*********************************************************************
```

Received 21 February 1979, 3 April 1979, and 26 October 1979.

Authors' addresses: U. Ascher, Computer Science Department, University of British Columbia, Vancouver, B.C. V6T 1W5, Canada; J. Christiansen and R.D. Russell, Mathematics Department, Simon Fraser University, Burnaby, B.C. V5A 1S6, Canada.

```
C
C     PURPOSE
C
C     SUBROUTINE COLSYS SOLVES A MULTI-POINT BOUNDARY VALUE
C     PROBLEM FOR A MIXED ORDER SYSTEM OF ODE-S GIVEN BY
C
C          (M(I))
C         U       =  F  ( X; Z(U(X)) )       I = 1, ... ,NCOMP
C          I          I
C
C                                           ALEFT .LT. X .LT. ARIGHT,
C
C
C         G  ( ZETA(J); Z(U(ZETA(J))) )= 0  J = 1, ... ,MSTAR
C          J
C                                    MSTAR=M(1)+M(2)+...+M(NCOMP),
C
C
C         WHERE                            T
C               U = (U , U , ... ,U       )  IS THE EXACT SOLUTION VECTOR
C                     1   2        NCOMP
C
C                (MI)
C               U      IS THE MI=M(I) TH  DERIVATIVE OF U
C                I                                       I
C
C                                (1)          (M1-1)        (MNCOMP-1)  T
C               Z(U(X)) = (U (X),U    (X),...,U     (X),...,U       (X))
C                           1     1            1             NCOMP
C
C               F (X,Z(U)) IS A (GENERALLY) NONLINEAR FUNCTION OF
C                I
C                          Z(U)=Z(U(X)).
C
C               G (ZETA(J);Z(U)) IS A (GENERALLY) NONLINEAR BOUNDARY
C                J
C                          CONDITION.
C
C        THE BOUNDARY POINTS SATISFY
C               ALEFT .LE. ZETA(1) .LE. .. .LE. ZETA(MSTAR) .LE. ARIGHT
C
C        THE ORDERS MI OF THE DIFFERENTIAL EQUATIONS SATISFY
C               M1 .LE. M2 .LE. ... .LE. MNCOMP .LE. 4.
C
C
C**********************************************************************
C
C     ***************     INPUT TO COLSYS     ***************
C
C     VARIABLES
C
C     NCOMP - NO. OF DIFFERENTIAL EQUATIONS   (NCOMP .LE. 20)
C
C     M(J) - ORDER OF THE J-TH DIFFERENTIAL EQUATION ( M(J).LE.M(J+1)
C            AND MSTAR = M(1) + ... + M(NCOMP) .LE. 40 )
C
C     ALEFT - LEFT END OF INTERVAL
C
C     ARIGHT - RIGHT END OF INTERVAL
C
C     ZETA(J) - J-TH SIDE CONDITION POINT (BOUNDARY POINT). MUST
C               HAVE  ZETA(J) .LE. ZETA(J+1)
C
C     IPAR - AN INTEGER ARRAY DIMENSIONED AT LEAST 11.
C            A LIST OF THE PARAMETERS IN IPAR AND THEIR MEANING FOLLOWS.
C            SOME PARAMETERS ARE RENAMED IN COLSYS, THESE NEW NAMES ARE
C            GIVEN IN PARENTHESES.
C
C     IPAR(1)      ( = NONLIN )
C             = 0 IF THE PROBLEM IS LINEAR
C             = 1 IF THE PROBLEM IS NONLINEAR
C
C     IPAR(2) = NO. OF COLLOCATION POINTS PER SUBINTERVAL  (= K )
C               WHERE M(NCOMP) .LT.  K .LE. 7 . IF IPAR(2)=0 THEN
C               COLSYS SETS  K = MAX ( M(NCOMP)+1, 5-M(NCOMP) )
C
```

```
C     IPAR(3) = NO. OF SUBINTERVALS IN THE INITIAL MESH  ( = N ).
C               IF IPAR(3) = 0 THEN COLSYS ARBITRARILY SETS N = 5.
C
C     IPAR(4) = NO. OF SOLUTION AND DERIVATIVE TOLERANCES.   ( = NTOL )
C               WE REQUIRE  0 .LT. NTOL .LE. MSTAR.
C
C     IPAR(5) = DIMENSION OF FSPACE.       ( = NDIMF )
C
C     IPAR(6) = DIMENSION OF ISPACE.       ( = NDIMI )
C
C     IPAR(7) -  OUTPUT CONTROL ( = IPRINT )
C              = -1 FOR FULL DIAGNOSTIC PRINTOUT
C              = 0 FOR SELECTED PRINTOUT
C              = 1 FOR NO PRINTOUT
C
C     IPAR(8)      ( = IREAD )
C             = 0 CAUSES COLSYS TO GENERATE A UNIFORM INITIAL MESH.
C             = 1 IF THE INITIAL MESH IS PROVIDED BY THE USER.   IT
C                 IS DEFINED IN FSPACE AS FOLLOWS:  THE MESH
C                 ALEFT=X(1).LT.X(2).LT. ... .LT.X(N).LT.X(N+1)=ARIGHT
C                 WILL OCCUPY  FSPACE(1), ..., FSPACE(N+1).  THE
C                 USER NEEDS TO SUPPLY ONLY THE INTERIOR MESH
C                 POINTS  FSPACE(J) = X(J), J = 2, ..., N.
C             = 2 IF THE INITIAL MESH IS SUPPLIED BY THE USER
C                 AS WITH IPAR(8)=1, AND IN ADDITION NO ADAPTIVE
C                 MESH SELECTION IS TO BE DONE.
C
C     IPAR(9)      ( = IGUESS )
C             = 0 IF NO INITIAL GUESS FOR THE SOLUTION IS
C                 PROVIDED.
C             = 1 IF AN INITIAL GUESS IS PROVIDED BY THE USER
C                 IN SUBROUTINE  SOLUTN.
C             = 2 IF AN INITIAL MESH AND APPROXIMATE SOLUTION
C                 COEFFICIENTS ARE PROVIDED BY THE USER IN  FSPACE.
C                 (THE FORMER AND NEW MESH ARE THE SAME).
C             = 3 IF A FORMER MESH AND AN APPROXIMATE SOLUTION
C                 COEFFICIENTS ARE PROVIDED BY THE USER IN FSPACE,
C                 AND THE NEW MESH IS TO BE TAKEN TWICE AS COARSE.
C             = 4 IF IN ADDITION TO A FORMER INITIAL MESH AND AN
C                 APPROXIMATE SOLUTION COEFFICIENTS, A NEW MESH
C                 IS PROVIDED IN FSPACE AS WELL.
C                 (SEE DESCRIPTION OF OUTPUT FOR FURTHER DETAILS
C                 ON IGUESS = 2, 3, AND 4. )
C
C     IPAR(10)= 0 IF THE PROBLEM IS REGULAR
C             = 1 IF THE FIRST RELAX FACTOR IS =RSTART, AND THE
C                 NONLINEAR ITERATION DOES NOT RELY ON PAST COVERGENCE
C                 (USE FOR AN EXTRA SENSITIVE NONLINEAR PROBLEM ONLY).
C             = 2 IF WE ARE TO RETURN IMMEDIATELY UPON  (A) TWO
C                 SUCCESSIVE NONCONVERGENCES, OR  (B) AFTER OBTAINING
C                 ERROR ESTIMATE FOR THE FIRST TIME.
C
C     IPAR(11)= NO. OF FIXED POINTS IN THE MESH OTHER THAN
C               ALEFT AND ARIGHT.  ( = NFXPNT , THE DIMENSION OF FIXPNT)
C
C     LTOL  -  AN ARRAY OF DIMENSION IPAR(4).  LTOL(J) = L  SPECIFIES
C              THAT THE J-TH TOLERANCE IN  TOL  CONTROLS THE ERROR
C              IN THE L-TH COMPONENT OF Z(U).    ALSO REQUIRE THAT
C              1.LE.LTOL(1).LT.LTOL(2).LT. ... .LT.LTOL(NTOL).LE.MSTAR
C
C     TOL    - AN ARRAY OF DIMENSION IPAR(4).  TOL(J) IS THE
C              ERROR TOLERANCE ON THE LTOL(J) -TH COMPONENT
C              OF Z(U).  THUS, THE CODE ATTEMPTS TO SATISFY
C              FOR J=1,...,NTOL  ON EACH SUBINTERVAL
C              ABS(Z(V)-Z(U))        .LE.  TOL(J)*Z(U)        +TOL(J)
C                          LTOL(J)                    LTOL(J)
C              IF V(X) IS THE APPROXIMATE SOLUTION VECTOR.
C
C     FIXPNT - AN ARRAY OF DIMENSION IPAR(11).    IT CONTAINS
C              THE POINTS, OTHER THAN ALEFT AND ARIGHT, WHICH
C              ARE TO BE INCLUDED IN EVERY MESH.
C
C     ISPACE - AN INTEGER WORK ARRAY OF DIMENSION IPAR(6).
C              ITS SIZE PROVIDES A CONSTRAINT ON NMAX,
C              THE MAXIMUM NUMBER OF SUBINTERVALS.  CHOOSE
C              IPAR(6) ACCORDING TO THE FORMULA
```

```
C                       IPAR(6)  .GE.  NMAX*NSIZEI
C                 WHERE
C                       NSIZEI = 3 + KDM - NREC
C                 WITH
C                       KDM = KD + MSTAR  ;  KD = K * NCOMP ;
C                       NREC = NO. OF RIGHT END BOUNDARY CONDITIONS.
C
C
C     FSPACE - A REAL WORK ARRAY OF DIMENSION IPAR(5).
C              ITS SIZE PROVIDES A CONSTRAINT ON NMAX.
C              CHOOSE IPAR(5) ACCORDING TO THE FORMULA
C                       IPAR(5)  .GE.  NMAX*NSIZEF
C                 WHERE
C                       NSIZEF = 4 + K + 2 * KD + (4+2*K) * MSTAR +
C                                (KDM-NREC) * (KDM+1).
C
C
C     IFLAG - THE MODE OF RETURN FROM COLSYS.
C           = 1 FOR NORMAL RETURN
C           = 0 IF THE COLLOCATION MATRIX IS SINGULAR.
C           =-1 IF THE EXPECTED NO. OF SUBINTERVALS EXCEEDS STORAGE
C               SPECIFICATIONS.
C           =-2 IF THE NONLINEAR ITERATION HAS NOT CONVERGED.
C           =-3 IF THERE IS AN INPUT DATA ERROR.
C
C
C*******************************************************************************
C
C     **********    USER SUPPLIED EXTERNAL SUBROUTINES    *******
C
C
C     FSUB  - NAME OF SUBROUTINE FOR EVALUATING F(X,Z(U(X))) =
C                                               T
C             (F ,...,F      )  AT A POINT X IN (ALEFT,ARIGHT).  IT
C               1       NCOMP
C             SHOULD HAVE THE HEADING
C
C                       SUBROUTINE FSUB (X , Z , F)
C
C             WHERE F IS THE VECTOR CONTAINING THE VALUE OF FI(X,Z(U))
C             IN THE I-TH COMPONENT AND                                 T
C                                       Z(U(X))=(Z(1),...,Z(MSTAR))
C             IS DEFINED AS ABOVE UNDER  PURPOSE .
C
C
C     DFSUB - NAME OF SUBROUTINE FOR EVALUATING THE JACOBIAN OF
C             F(X,Z(U)) AT A POINT X.  IT SHOULD HAVE THE HEADING
C
C                       SUBROUTINE DFSUB (X , Z , DF)
C
C             WHERE Z(U(X)) IS DEFINED AS FOR FSUB AND THE (NCOMP) BY
C             (MSTAR) ARRAY DF SHOULD BE FILLED BY THE PARTIAL DERIV-
C             ATIVES OF F, VIZ, FOR A PARTICULAR CALL ONE CALCULATES
C                             DF(I,J) = DFI / DZJ,  I=1,...,NCOMP
C                                                   J=1,...,MSTAR.
C
C     GSUB  - NAME OF SUBROUTINE FOR EVALUATING THE I-TH COMPONENT OF
C             G(X,Z(U(X))) = G (ZETA(I),Z(U(ZETA(I)))) AT A POINT X =
C                             I
C             ZETA(I) WHERE 1.LE.I.LE.MSTAR.  IT SHOULD HAVE THE HEADING
C
C                       SUBROUTINE GSUB (I , Z , G)
C
C             WHERE Z(U) IS AS FOR FSUB, AND I AND G=G  ARE AS ABOVE.
C                                                   I
C             NOTE THAT IN CONTRAST TO F IN  FSUB , HERE
C             ONLY ONE VALUE PER CALL IS RETURNED IN G.
C
C
C     DGSUB - NAME OF SUBROUTINE FOR EVALUATING THE I-TH ROW OF
C             THE JACOBIAN OF G(X,U(X)).  IT SHOULD HAVE THE HEADING
C
C                       SUBROUTINE DGSUB (I , Z , DG)
C
C             WHERE Z(U) IS AS FOR FSUB, I AS FOR GSUB AND THE MSTAR-
C             VECTOR DG SHOULD BE FILLED WITH THE PARTIAL DERIVATIVES
```

```
C             OF G, VIZ, FOR A PARTICULAR CALL ONE CALCULATES
C                   DG(I,J) = DGI / DZJ       J=1,...,MSTAR.
C
C
C     SOLUTN- NAME OF SUBROUTINE TO EVALUATE THE INITIAL
C             APPROXIMATION FOR  Z(U(X)) AND FOR DMVAL(U(X))= VECTOR
C             OF THE MJ-TH DERIVATIVES OF U(X).  IT SHOULD HAVE THE
C             HEADING
C
C                       SUBROUTINE SOLUTN (X , Z , DMVAL)
C
C             NOTE THAT THIS SUBROUTINE IS NEEDED ONLY IF USING
C             IPAR(9) = 1, AND THEN ALL  MSTAR  COMPONENTS OF Z
C             AND  NCOMP  COMPONENTS OF  DMVAL  SHOULD BE SPECIFIED
C             FOR ANY X,  ALEFT .LE. X .LE. ARIGHT .
C
C
C*********************************************************************
C
C     ***************     OUTPUT FROM COLSYS    ***************
C
C
C     UPON RETURN FROM   COLSYS , THE USER MAY PRODUCE THE
C     SOLUTION VECTOR  Z( U(X) )  AT A POINT X, ALEFT.LE.X.LE.ARIGHT
C     BY CALLING :
C
C           CALL APPSLN (X, Z, FSPACE, ISPACE)
C
C     THIS SETS UP A STANDARD CALL TO  APPROX . FOR A MORE
C     EFFICIENT OR SOPHISTICATED RETRIEVAL OF THE SOLUTION
C     VALUES, CALL  APPROX  DIRECTLY (SEE DOCUMENTATION IN
C     APPROX - THE PARAMETERS NEEDED IN THE CALL TO  APPROX
C     BY THE USER ARE SAVED IN  ISPACE  AND  FSPACE  BEFORE
C     COLSYS  RETURNS).
C
C     IN ORDER TO SAVE THE COEFFICIENTS OF THE SOLUTION FOR LATER
C     REFERENCE,     ISPACE(1), ..., ISPACE(7+MSTAR)   AND
C                    FSPACE(1), ..., FSPACE(ISPACE(7))  SHOULD BE
C     SAVED, SINCE THESE ARE USED IN THE CALL TO  APPSLN (APPROX).
C
C     ONE CAN ALSO USE THE FORMERLY OBTAINED APPROXIMATE
C     SOLUTION AS A FIRST APPROXIMATION FOR THE NONLINEAR ITERATION
C     ON A NEW PROBLEM (E.G. FOR CONTINUATION PURPOSES).   THIS
C     INVOLVES USING IGUESS = 2, 3, OR 4, AS FOLLOWS:
C
C     FOR IGUESS= 2 OR 3, THE USER SHOULD PUT THE ABOVE SAVED
C     VALUES BACK INTO FSPACE(1),...,FSPACE(ISPACE(6)).
C     THE SIZE OF THE FORMER MESH, NOLD, IS PROVIDED IN IPAR(3).  IF
C     IGUESS=2 THEN THE SIZE OF THE NEW MESH, N, IS TAKEN TO BE =NOLD.
C     IF IGUESS=3 THEN N := NOLD/2 AND THE NEW MESH IS TO BE TWICE AS
C     COARSE.
C     FOR IGUESS=4, PUT N IN IPAR(3) AND NOLD IN ISPACE(1).  THE
C     VALUES OF THE FORMER SOLUTION, SAVED AS DESCRIBED ABOVE,
C     SHOULD BE PUT INTO FSPACE(N+2),...,FSPACE(ISPACE(6)+N+1),   AND
C     A NEW MESH UNRELATED TO THE FORMER ONE IS PRESCRIBED IN
C     FSPACE(1),...,FSPACE(N+1).
C
C
C*********************************************************************
C
C     ***************      PACKAGE SUBROUTINES      ***************
C
C     THE FOLLOWING DESCRIPTION GIVES A BRIEF OVERVIEW OF HOW THE
C     PROCEDURE IS BROKEN DOWN INTO THE SUBROUTINES WHICH MAKE UP
C     THE PACKAGE CALLED  COLSYS . FOR FURTHER DETAILS THE
C     USER SHOULD REFER TO DOCUMENTATION IN THE VARIOUS SUBROUTINES
C     AND TO THE REFERENCES CITED ABOVE.
C
C     THE SUBROUTINES FALL INTO FOUR GROUPS:
C
C PART 1 - THE MAIN STORAGE ALLOCATION AND PROGRAM CONTROL SUBROUTINES.
C
C     COLSYS - TESTS INPUT VALUES, DOES INITIALIZATION AND BREAKS UP
C              THE WORK AREAS, FSPACE AND ISPACE, INTO THE ARRAYS
C              USED BY THE PROGRAM.
C
```

```
C     CONTRL - IS THE ACTUAL DRIVER OF THE PACKAGE.  THIS ROUTINE
C              CONTAINS THE STRATEGY FOR NONLINEAR PROBLEMS.
C
C
C PART 2 - MESH SELECTION AND ERROR ESTIMATION SUBROUTINES
C
C     CONSTS - IS CALLED ONCE BY  COLSYS  TO INITIALIZE CONSTANTS
C              WHICH ARE USED FOR ERROR ESTIMATION AND MESH SELECTION.
C
C     NEWMSH - GENERATES MESHES.  IT CONTAINS THE TEST TO DECIDE
C              WHETHER OR NOT TO REDISTRIBUTE A MESH.
C
C     ERRCHK - PRODUCES ERROR ESTIMATES AND CHECKS AGAINST THE
C              TOLERANCES AT EACH SUBINTERVAL
C
C
C PART 3 - COLLOCATION SYSTEM SET-UP SUBROUTINES
C
C     LSYSLV - CONTROLS THE SET-UP AND SOLUTION OF THE LINEAR
C              ALGEBRAIC SYSTEMS OF COLLOCATION EQUATIONS WHICH
C              ARISE AT EACH NEWTON ITERATION.
C
C     BLDBLK - IS USED BY LSYSLV TO SET UP THE EQUATION(S) ASSOCIATED
C              WITH A SIDE CONDITION POINT OR A COLLOCATION POINT.
C
C
C PART 4 - B-SPLINE SUBROUTINES
C
C     APPSLN - SETS UP A STANDARD CALL TO  APPROX
C
C     APPROX - EVALUATES A PIECEWISE POLYNOMIAL SOLUTION.
C
C     BSPFIX - EVALUATES THE MESH INDEPENDENT B-SPLINES
C              (I.E.  THE FIXED B-SPLINES)
C
C     BSPVAR - EVALUATES THE MESH DEPENDENT B-SPLINES (I.E.  THE
C              VARYING B-SPLINES)
C
C     BSPDER - GENERATES VALUES FOR THE DERIVATIVES NEEDED TO SET
C              UP THE COLLOCATION EQUATIONS.
C
C     APPDIF - GENERATES A DIVIDED DIFFERENCE TABLE FROM THE B-SPLINE
C              COEFFICIENTS FOR A COLLOCATION SOLUTION.  THE TABLE
C              IS USED IN  APPROX
C
C     HORDER - EVALUATES THE HIGHEST ORDER DERIVATIVES OF THE
C              CURRENT COLLOCATION SOLUTION USED FOR MESH REFINEMENT.
C
C
C     TO SOLVE THE LINEAR SYSTEMS OF COLLOCATION EQUATIONS
C     CONSTRUCTED IN PART 3,  COLSYS  USES THE PACKAGE
C     SOLVEBLOK  OF DE BOOR - WEISS (TO APPEAR IN TOMS).
```

ALGORITHM 570
LOPSI: A Simultaneous Iteration Algorithm for Real Matrices [F2]

WILLIAM J. STEWART
North Carolina State University
and
ALAN JENNINGS
Queen's University, Belfast, Northern Ireland

Key Words and Phrases: eigenvalues, eigenvectors, simultaneous iteration, real unsymmetric matrices, large sparse matrices
CR Categories: 5.14
Language: FORTRAN

DESCRIPTION

The algorithm given here is a complement to [1] where the description, test results, and references are given.

REFERENCES

1. STEWART, W.J., AND JENNINGS, A. A simultaneous iteration algorithm for real matrices. *ACM Trans. Math. Softw. 7*, 2 (June 1981), 184–198.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service .]

```
      SUBROUTINE LOPSI(N, A, IA, IROW, ICOL, LR, IR, M, IS, TOL, ITMAX, LMAX, U, IU,
     1            V, W, G, H, B, P, IQ, INT, X, Y, ERR, ITTOT, INACT, IFAIL)
C
C
C
C        **  LOPSI IS A SIMULTANEOUS ITERATION ALGORITHM WHICH
C            DETERMINES APPROXIMATIONS TO RIGHT OR LEFT EIGEN-
C              VECTORS CORRESPONDING TO THE DOMINANT SET OF
C              EIGENVALUES OF A REAL UNSYMMETRIC MATRIX A.          **
C
C                   BY  WILLIAM J. STEWART  AND  ALAN JENNINGS
C
C
C
C      FORMAL PARAMETER LIST.
```

Received 12 December 1977, 12 June 1979, and 14 April 1980.

Author's addresses: W. J. Stewart, Department of Computer Science, North Carolina State University, P.O. Box 5972, Raleigh, NC 27650; A. Jennings, Department of Civil Engineering, Queen's University, Belfast, BT7 1NN Northern Ireland.

```
C     **********
C
C
C     INPUT PARAMETERS   (MUST BE SET BEFORE ENTRY AND ARE UNALTERED
C     ***********        BY THE EXECUTION OF THE ALGORITHM)
C
C
C     N      AN INTEGER QUANTITY, THE ORDER OF THE MATRIX  A  FOR WHICH
C            THE PARTIAL EIGENSOLUTION IS TO BE OBTAINED.
C
C     A      A ONE-DIMENSIONAL REAL ARRAY OF LENGTH AT LEAST EQUAL
C            TO  IA.  IT CONTAINS IN ARBITRARY ORDER THE NON-ZERO
C            ELEMENTS OF THE MATRIX A.
C
C     IA     AN INTEGER QUANTITY,  THE NUMBER OF NON-ZERO ELEMENTS IN
C            THE MATRIX A.
C
C     IROW   A ONE-DIMENSIONAL INTEGER ARRAY OF LENGTH AT LEAST EQUAL TO
C            IA.  THE I-TH COMPONENT DENOTES THE ROW POSITION OF THE
C            NON-ZERO ELEMENT STORED IN POSITION I OF ARRAY A.
C
C     ICOL   A ONE-DIMENSIONAL INTEGER ARRAY OF LENGTH AT LEAST EQUAL TO
C            IA.  THE I-TH COMPONENT DENOTES THE COLUMN POSITION OF THE
C            NON-ZERO ELEMENT STORED IN POSITION I OF ARRAY A.
C
C     LR     AN INTEGER QUANTITY. IF LR=1 ON ENTRY, THEN THE RIGHT EIGEN-
C            VECTORS CORRESPONDING TO THE DOMINANT EIGENVALUES WILL BE
C            OBTAINED. OTHERWISE THE LEFT SET WILL BE OBTAINED.
C
C     IR     AN INTEGER QUANTITY, THE NO. OF VECTORS REQUIRED ACCURATELY.
C
C     M      AN INTEGER QUANTITY, THE NO. OF TRIAL VECTORS EMPLOYED.
C            M SHOULD NORMALLY BE CHOSEN SO THAT M >= IR+2
C
C     IS     AN INTEGER QUANTITY, THE NUMBER OF TRIAL VECTORS FOR WHICH
C            INITIAL APPROXIMATIONS ARE SUPPLIED BY THE USER.
C
C     TOL    A REAL QUANTITY, THE TOLERANCE DEMANDED OF THE EIGENVECTORS
C            CORRESPONDING TO THE  IR  DOMINANT EIGENVALUES.
C            THE EIGENVALUES ARE NORMALIZED TO HAVE LARGEST COMPONENT
C            UNITY AND "LOPSI" AIMS FOR EACH COMPONENT TO BE IN ERROR
C            BY LESS THAN  TOL.  IN GENERAL IT CAN BE EXPECTED THAT
C            THE CORRESPONDING EIGENVALUES HAVE A RELATIVE ERROR
C            LESS THAN TOL.
C
C     ITMAX  AN INTEGER QUANTITY, AN UPPER BOUND ON THE NUMBER OF MATRIX
C            BY VECTOR MULTIPLICATIONS TO BE PERFORMED
C
C     LMAX   AN INTEGER QUANTITY, THE MAXIMUM NUMBER OF PREMULTIPLIC-
C            ATIONS PERMITTED BETWEEN TWO REORIENTATIONS
C            THE VALUE LMAX = 10 SHOULD BE ADEQUATE FOR MOST PROBLEMS
C            HOWEVER SET LMAX = 1 WHEN EXPECTING FAST CONVERGENCE
C            FOR FURTHER INFORMATION SEE SECTION 3 OF THE PAPER
C     IU     AN INTEGER QUANTITY, THE DECLARED FIRST DIMENSION OF
C            ARRAYS U, V AND W.
C
C     IG     AN INTEGER QUANTITY, THE DECLARED FIRST DIMENSION
C            OF WORK ARRAYS G, H, B, P.
C
C
C      INPUT AND OUTPUT PARAMETER
C      **************************
C
C
C     U      A TWO-DIMENSIONAL REAL ARRAY OF AT LEAST (N, M) ELEMENTS. ON
C            ENTRY IT CONTAINS USER SUPPLIED APPROXIMATIONS TO THE EIGEN-
C            VECTORS CORRESPONDING TO THE  IS  DOMINANT EIGENVALUES.
C            ON EXIT IT CONTAINS THE FINAL ESTIMATES TO THE EIGENVECTORS
C            CORRESPONDING TO THE  M  DOMINANT EIGENVALUES.
C
C
C     OUTPUT PARAMETERS
C     *****************
C
C
```

```
C     X, Y  ONE-DIMENSIONAL REAL ARRAYS OF LENGTH AT LEAST EQUAL TO M.
C           ON EXIT THE REAL AND IMAGINARY PARTS OF THE J-TH EIGENVALUE
C           PREDICTION ARE STORED IN X(J) AND Y(J) RESPECTIVELY.
C
C     ERR   A ONE-DIMENSIONAL REAL ARRAY OF AT LEAST M ELEMENTS. ON EXIT
C           ERR(J) DENOTES THE ESTIMATED ACCURACY OF THE J-TH VECTOR.
C           ERROR ESTIMATES, AS MEASURED BY THE MAXIMUM OF THE ABSOLUTE
C           DIFFERENCES IN THE COMPONENTS OF SUCCESSIVE APPROXIMATIONS,
C           ARE GIVEN FOR ALL  M  VECTORS INCLUDING THOSE THAT HAVE NOT
C           SATISFIED THE CONVERGENCE CRITERIA SPECIFIED UNDER  TOL.
C
C     ITTOT AN INTEGER QUANTITY WHICH ON EXIT DENOTES THE
C           NUMBER OF PREMULTIPLICATIONS PERFORMED.
C
C     INACT AN INTEGER QUANTITY WHICH ON EXIT DENOTES THE
C           NUMBER OF INTERACTION ANALYSES PERFORMED.
C
C     IFAIL AN INTEGER QUANTITY. ON EXIT IFAIL=0 IF FIRST IR VECTORS
C           HAVE SATISFIED THE CONVERGENCE CRITERIA, OTHERWISE IFAIL=1.
C
C
C     WORK ARRAYS
C     ***********
C
C
C     V,W   TWO-DIMENSIONAL REAL WORK ARRAYS OF AT LEAST (N, M) ELEMENTS
C
C     G,H,  FOUR TWO-DIMENSIONAL REAL WORK ARRAYS
C     B,P   OF AT LEAST (M, M) ELEMENTS EACH.
C
C     INT   A ONE-DIMENSIONAL INTEGER WORK ARRAY OF AT LEAST M ELEMENTS.
C
C
C
      DOUBLE PRECISION A,B,DIFF1,DIFF2,DLAM,DLAMM,DLAMR,DLAM1,EMAX
      DOUBLE PRECISION ERR,FMAX,G,H,P,SIGMA,SUM,TOIT,TOL,TOT,TOT1,TOT2
      DOUBLE PRECISION U,V,W,X,Y,EPMACH,EPS
      DIMENSION A(IA),IROW(IA),ICOL(IA),U(IU,M),V(IU,M),W(IU,M)
      DIMENSION G(IG,M),H(IG,M),B(IG,M),P(IG,M)
      DIMENSION X(M),Y(M),INT(M),ERR(M)
```

ALGORITHM 571
Statistics for von Mises' and Fisher's Distributions of Directions: $I_1(x)/I_0(x)$, $I_{1.5}(x)/I_{0.5}(x)$, and Their Inverses [S14]

GEOFFREY W. HILL
CSIRO, Australia

Key Words and Phrases: direction statistics, von Mises distribution, Fisher distribution, modified Bessel function ratio, continued fraction, function inversion, Newton–Raphson
CR Categories: 5.5, 5.12
Language: FORTRAN

DESCRIPTION

Two pairs of FORTRAN functions provide for evaluation of the ratio of modified Bessel functions of the first kind, $A(\kappa) = I_1(\kappa)/I_0(\kappa)$ or $B(\kappa) = I_{1.5}(\kappa)/I_{0.5}(\kappa) = \coth \kappa - 1/\kappa$ or their inverses, $A^{-1}(R)$, $B^{-1}(R)$. $A(\kappa)$ is the expected value of the mean modulus $\bar{R}$ of random unit vectors in two dimensions, distributed with concentration parameter κ in von Mises' [5] distribution of directions on a circle, whereas $B(\kappa) = E(\bar{R})$ for Fisher's [1] distribution of three-dimensional directions on a sphere. The inverse functions, $A^{-1}(\bar{R})$ and $B^{-1}(\bar{R})$ are corresponding maximum likelihood estimators of κ for observed $\bar{R}$.

Other versions of each function are fairly readily achieved, which provide some choice of precision level in the real valued result, summarized in Table I.

The function BESRAT is illustrated as a 9.3S version, which selects between continued fractions detailed in [3] for backward recursive evaluation from a depth determined by parameter values C1 and C2, assigned with CX to DATA constants as indicated in comment cards. Within the tested range of precision the values specified for these parameters prevent occurrence of zero divisors in evaluation of the continued fractions. A version with S decimal digit precision assignable-at-call (up to limitations imposed by processor precision) can be obtained by changing the first card to read FUNCTION BESRAT(V, S), by deleting C from column 1 of cards defining C1, C2, and CX, and by suppressing the DATA card.

The function VKAPPA achieves 8S results by means of different initial approximations and up to two Newton–Raphson improvements adapted for several subintervals of the range of valid arguments. The Newton–Raphson steps call on BESRAT in a 9.3S version to compensate for some loss of precision for R near 1, such as occurs in Algorithm AS81 [6]. None of the five expressions

Received 28 November 1978, 5 September 1979, and 24 October 1979.

Author's address: Division of Mineral Chemistry, CSIRO, Port Melbourne, Australia 3207.

Table I

FORTRAN Function	Argument[a]	Result	Precision[b]
BESRAT(V)	$0 \le \kappa = \lvert V \rvert$	$A(\kappa)$ in (0, 1)	9.3 (5–14)
VKAPPA(R)	$0 \le R < 1$	$A^{-1}(R) \ge 0$	8 (5)
SPHERR(CAPPA)	$0 \le \kappa = \lvert$CAPPA$\rvert$	$B(\kappa)$ in (0, 1)	13 (5–20)
CAPPA3(R)	$0 \le R < 1$	$B^{-1}(R) \ge 0$	16 (8, 25)

[a] Definition of κ as the modulus of the actual parameter value avoids an error condition, whereas for invalid argument $R < 0$ or $R \ge 1$ the value -1.0 is returned as an error signal result.

[b] Precision is quoted as the least number of correct significant decimal digits in test results on a CDC 7600 of the FORTRAN version illustrated (range of precision tested is noted in parentheses). It is assumed that argument values are represented exactly and that the result precision is not appreciably limited by that of the processor.

occurring as divisors in the code can become zero for arguments within the relevant subintervals, so that processor arithmetic overflow cannot occur.

For analysis of observational data, less precise versions may be adequate [2], such as a fast 5*S* version of VKAPPA, obtained by assigning the value 0.74 to V1 and 0.89 to V2 in the DATA statement and by replacing the two conditional statements before label 30 by columns 29 to 72 of either to leave one unconditional Newton–Raphson call of a 6*S* version of BESRAT. In any case, the backward recursive evaluation of the continued fractions reduces accumulation of roundoff error and enables a substantially faster and more compact code than previously published procedures [2, 6], which involve some ten or more evaluations of numerator and denominator of the Bessel function ratio.

The function SPHERR determines $B(\kappa)$ either by backward recursive evaluation of the continued fraction form of $\coth \kappa - 1/\kappa$ or, if $\kappa > 1$, by means of the exponential form of coth, as outlined in [3]. Unnecessary evaluation of the exponential function and possible overflow for large argument is avoided by evaluating $B(\kappa)$ as $1 - 1/\kappa$ for large values of $\kappa >$ BIGX $= 1.1513 \times S + 0.4$, which ensures S decimal digit precision in the result. The depth of recursion is also simply determined as a function of desired precision S, as indicated in comment cards defining DATA constants. A version with precision assignable-at-call may be obtained by including S as a formal parameter and replacing the DATA statement by the assignments SON4 $= S/4$ and BIGX $= 1.1513 \times S + 0.4$.

The function CAPPA3 achieves 8*S* precision in the result $B^{-1}(R)$ by means of inverses of the continued fraction and exponential forms of $\coth \kappa - 1/\kappa$, empirically extended over different subintervals of the argument [3]. Unnecessary evaluation of the exponential function and possible overflow for argument R near 1.0 are avoided by accepting $1/(1 - R)$ as the result, if that value exceeds BIGX $= 1.1513 \times S + 0.4$. Of course, the effective precision of the argument and therefore that of the result are limited by that of the difference, $1 - R$.

Precision of the initial approximation may be extended by Newton–Raphson inversion of $B(\kappa)$, using SPHERR(−) to provide the Bessel function ratio to a precision of $S + 1$ decimals to allow for loss of up to one decimal digit precision in the improvement. A version with precision up to 8*S* is obtained by deleting statements between label 20 and 30 of the version below, which caters for precision up to 16*S*. Precision could be extended to 25*S* by repeating the statement preceding label 30 or 32*S* by repeating the three statements preceding label 30, changing the DATA constants W1, W2 from 10.0, 0.01 to 20.0 and 0.001, respectively, and adjusting the precision of the auxiliary procedure SPHERR(−) accordingly.

Test programs on a CDC 7600 checked precision claims using single-precision (48*B*) versions of BESRAT and VKAPPA and double-precision (96*B*) versions of SPHERR and CAPPA3. Tabulated values checked against tables to 5*D* [5] identified some errors in published tables [4], but otherwise confirmed the validity of the functions BESRAT, VKAPPA, SPHERR, and CAPPA3.

REFERENCES

1. FISHER, R.A. Dispersion on a sphere. *Proc. Roy. Soc. London, A217* (1953), 295–305.
2. HARVEY, P.K., AND FERGUSON, C.C. On testing orientation data for goodness-of-fit to a von Mises distribution. *Comput. Geosci. 2* (1976), 261–268.
3. HILL, G.W. Evaluation and inversion of the ratios of modified Bessel functions, $I_1(x)/I_0(x)$ and $I_{1.5}(x)/I_{0.5}(x)$. *ACM Trans. Math. Softw. 7*, 2 (June 1981), 199–208.
4. HILL, G.W. Table errata: #561 Edward Batschelet (1965) and #562 K.V. Mardia (1972). *Math. Comput. 33* (1979), 845.
5. MARDIA, K.V. *Statistics of Directional Data.* Academic Press, London, 1972.
6. MARDIA, K.V., AND ZEMROCH, P.J. Algorithm AS81. Circular statistics. *Appl. Stat. 24*, 1 (1975), 147–150.

ALGORITHM

[The complete listing is printed here and is available from the ACM Algorithms Distribution Service .]

```
      FUNCTION BESRAT(V)
C -----------------------------------------------------------------
C RETURNS BESRAT = A(K) FOR K = ABS(V), WHERE A(K) IS THE EXPECTED
C MODULUS OF THE MEAN VECTOR SUM OF UNIT VECTORS SAMPLED FROM THE
C VON MISES DISTRIBUTION OF DIRECTIONS IN 2D WITH PARAMETER = K.
C A(V) = THE RATIO OF MODIFIED BESSEL FUNCTIONS OF THE FIRST KIND
C OF ORDERS 1 AND 0, I.E., A(V) = I1(V)/I0(V).
C -----------------------------------------------------------------
C
C  ADJUST TO S DECIMAL DIGIT PRECISION BY SETTING DATA CONSTANTS -
C     C1 = (S+9.0-8.0/S)*0.0351
C     C2 = ((S-5.0)**3/180.0+S-5.0)/10.0
C     CX = S*0.5 + 11.0
C  FOR S IN RANGE (5,14).   THUS FOR S = 9.3 :
      DATA C1/0.613/, C2/0.475/, CX/15.65/
C
      Y = 0.0
      X = ABS(V)
      IF (X.GT.CX) GO TO 20
C
C  FOR SMALL X, RATIO = X/(2+X*X/(4+X*X/(6+X*X/(8+ ... )))
         N = INT((X+16.0-16.0/(X+C1+0.75))*C1)
         X = X*0.5
         XX = X*X
         DO 10 J = 1, N
10       Y = XX/(FLOAT(N-J+2)+Y)
         BESRAT = X/(1.0+Y)
         RETURN
C
C  FOR LARGE X, RATIO = 1-2/(4X-1-1/(4X/3-2-1/(4X/5-2- ... )))
20    N = INT((68.0/X+1.0)*C2) + 1
      X = X*4.0
      XX = FLOAT(N*2+1)
      DO 30 J = 1, N
         Y = XX/((-2.0-Y)*XX+X)
30       XX = XX - 2.0
      BESRAT = 1.0-2.0/(X-1.0-Y)
      RETURN
      END



      FUNCTION VKAPPA(R)
C -----------------------------------------------------------------
C RETURNS VKAPPA = THE MAXIMUM LIKELIHOOD ESTIMATE OF 'KAPPA', THE
C CONCENTRATION PARAMETER OF VON MISES' DISTRIBUTION OF DIRECTIONS
C IN 2 DIMENSIONS, CORRESPONDING TO A SAMPLE MEAN VECTOR MODULUS R.
C VKAPPA = K(A), THE INVERSE FUNCTION OF  A(K) = RATIO OF MODIFIED
C BESSEL FUNCTIONS OF THE FIRST KIND, VIZ., A(K) = I1(K)/I0(K).
C -----------------------------------------------------------------
C
C  FOR 8S (SIGNIFICANT DECIMAL DIGITS) PRECISION AUXILIARY ROUTINE
C  FUNCTION BESRAT(V) MUST BE SET TO AT LEAST 9.3S
      DATA V1/0.642/, V2/0.95/
      A = R
      S = -1.0
C
```

```
C    ERROR SIGNAL: VALUE -1.0 RETURNED IF ARGUMENT -VE OR 1.0 OR MORE.
       IF (A.LT.0.0 .OR. A.GE.1.0) GO TO 30
       Y = 2.0/(1.0-A)
       IF (A.GT.0.85) GO TO 10
C
C    FOR R BELOW 0.85 USE ADJUSTED INVERSE TAYLOR SERIES.
          X = A*A
          S = (((A-5.6076)*A+5.0797)*A-4.6494)*Y*X - 1.0
          S = ((((S*X+15.0)*X+60.0)*X/360.0+1.0)*X-2.0)*A/(X-1.0)
          IF (V1-A) 20,20,30
C
C    FOR R ABOVE 0.85 USE CONTINUED FRACTION APPROXIMATION.
  10   IF (A.GE.0.95) X = 32.0/(120.0*A-131.5+Y)
       IF (A.LT.0.95) X = (-2326.0*A+4317.5526)*A-2001.035224
       S = (Y+1.0+3.0/(Y-5.0-12.0/(Y-10.0-X)))*0.25
       IF (A.GE.V2) GO TO 30
C
C    FOR R IN (0.642,0.95) APPLY NEWTON-RAPHSON, TWICE IF R IN
C    (0.75,0.875), FOR 8S PRECISION, USING APPROXIMATE DERIVATIVE -
  20          Y = ((0.00048*Y-0.1589)*Y+0.744)*Y - 4.2932
              IF (A.LE.0.875) S = (BESRAT(S)-A)*Y + S
              IF (A.GE.0.75)  S = (BESRAT(S)-A)*Y + S
  30   VKAPPA = S
       RETURN
       END


       FUNCTION SPHERR(CAPPA)
C ---------------------------------------------------------------
C RETURNS SPHERR = B(K) FOR K = ABS(CAPPA).  B(K) IS THE EXPECTED
C MODULUS OF THE MEAN VECTOR SUM OF UNIT VECTORS SAMPLED FROM THE
C FISHER DISTRIBUTION OF DIRECTIONS IN 3D WITH PARAMETER = CAPPA.
C B(K) = THE RATIO OF MODIFIED BESSEL FUNCTIONS OF THE FIRST KIND
C OF ORDERS 3/2 AND 1/2, EQUIVALENT TO COTH(K) - 1/K.
C ---------------------------------------------------------------
C
C FOR S DECIMAL DIGIT PRECISION AND TO AVOID EXPONENTIAL OVERFLOW
C SET SON4 = S/4 AND BIGX = 1.1513*S+0.4, E.G., FOR 48B = 14.45S;
       DATA SON4/3.61/, BIGX/17.0/
C
       T = 0.0
       X = ABS(CAPPA)
       IF (X.GT.1.0) GO TO 20
C
C FOR SMALL X EVALUATE GAUSS CONTINUED FRACTION X/(3+X*X/(5+...))
C UP FROM J-TH LEVEL, WHERE N=2*J+1 YIELDS S DECIMALS PRECISION.
          N = INT((X+0.88)*SON4)*2 + 5
          XX = X*X
  10        T = XX/(FLOAT(N)+T)
            N = N - 2
            IF (N.GT.3) GO TO 10
          SPHERR = X/(T+3.0)
          RETURN
C
C FOR LARGE X USE EXPONENTIAL FORM OF COTH(X) - 1/X
  20   IF (X.LT.BIGX) T = 2.0/(EXP(2.0*X)-1.0)
       SPHERR = T + 1.0 - 1.0/X
       RETURN
       END


       FUNCTION CAPPA3(R)
C ---------------------------------------------------------------
C RETURNS CAPPA3 = THE MAXIMUM LIKELIHOOD ESTIMATE OF 'KAPPA', THE
C CONCENTRATION PARAMETER OF THE FISHER DISTRIBUTION OF DIRECTIONS
C IN 3 DIMENSIONS, CORRESPONDING TO A SAMPLE MEAN VECTOR MODULUS R.
C CAPPA3 = K(B), THE INVERSE FUNCTION OF  B(K) = RATIO OF MODIFIED
C BESSEL FUNCTIONS OF THE FIRST KIND OF ORDERS 3/2 AND 1/2.
C ---------------------------------------------------------------
C
C FOR PRECISION UP TO 8S (SIGNIFICANT DECIMAL DIGITS) OMIT THREE
C LINES FOLLOWING STATEMENT LABELED 20.  FOR GREATER PRECISION UP
C TO 16S THE AUXILIARY SUBPROGRAM, FUNCTION SPHERR(X), IS NEEDED
C WITH PRECISION ONE DECIMAL DIGIT GREATER THAN FOR CAPPA3(R).
C TO AVOID EXPONENTIAL OVERFLOW SET BIGX = 1.1513*S + 0.4; FOR 48B
       DATA BIGX /17.0/, W1/10.0/, W2/0.01/
       Y = R
```

```
      X = -1.0
C
C ERROR SIGNAL: VALUE -1.0 RETURNED IF ARGUMENT -VE OR 1.0 OR MORE
      IF (Y.LT.0.0 .OR. Y.GE.1.0) GO TO 30
      IF (Y.LT.0.5) GO TO 10
C
C FOR LARGE R APPROX. INVERSE OF R = COTH(K) - 1/K. (YIELDS 8.1S)
        X = 1.0/(1.0-Y)
        IF (X.GT.BIGX) GO TO 30
        S = 2.0*X
        T = EXP(S) - S*S - 1.0
        IF (Y.LT.0.8) T = (((0.00254*S-0.071042)*S+0.6943388)*S
                          -2.3816184)*S + 0.1508478 - 0.14789*Y + T
        X = T*X/(T+S)
        IF (X-W1) 20,20,30
C
C FOR SMALL R USE INVERSE GAUSS CONTINUED FRACTION. (YIELDS 8.4S)
  10  X = 3.0*Y
      S = X*X
      T = 12.375
      IF (X.GT.0.7) T = ((5.0*X-14.74)*X+16.5198)*X+6.2762
      X = X/(1.0-S/(15.0-4.0*S/(7.0-S/(5.0-S/T))))
C
C ONE STAGE NEWTON-RAPHSON INVERSION DOUBLES PRECISION.
  20  IF (X.LT.W2) GO TO 30
      S = EXP(X)
      S = S*2.0/(S*S-1.0)
      X = X + (Y-SPHERR(X))/(1.0/(X*X)-S*S)
C
  30  CAPPA3 = X
      RETURN
      END
```

ALGORITHM 572
Solution of the Helmholtz Equation for the Dirichlet Problem on General Bounded Three-Dimensional Regions [D3]

DIANNE P. O'LEARY
University of Maryland
and
OLOF WIDLUND
New York University

Key Words and Phrases: Helmholtz equation, capacitance matrix, fast Poisson solvers, conjugate gradients
CR Categories: 5.17
Language: FORTRAN

DESCRIPTION

This algorithm provides an approximate solution to the Helmholtz equation

$$-\Delta u + cu = g \qquad \text{in } \Omega$$

with a Dirichlet boundary condition

$$u = f \qquad \text{on } \Gamma, \text{ the boundary of } \Omega.$$

Here Ω, a three-dimensional bounded region, c, an arbitrary real constant (positive, negative, or zero), and the functions f and g are specified by the user. The Laplace operator Δ is in Cartesian coordinates.

A second-order accurate finite-difference method is used to discretize the Helmholtz equation. The resulting linear system of equations is reduced to a capacitance matrix equation that is solved approximately by a conjugate gradient method. We sketch the basic ideas below, but a detailed discussion of this and similar methods can be found in [1].

To perform the discretization, the region Ω is embedded in a cube and a uniform rectangular finite-difference grid is imposed. A simple seven-point difference approximation is used for all mesh points except those that are in Ω and are near the boundary Γ. For these boundary neighbors, second-order accurate equations incorporating the boundary data are used. The resulting difference

Received 29 January 1979, 22 October 1979, and 21 December 1979.

The work of D. P. O'Leary was supported by National Science Foundation Grant MCS 76-06595, while at the University of Michigan; the work of O. Widlund was supported by the Energy Research and Development Agency under Contract EY-76-C-02-3077.
Authors' addresses: D. P. O'Leary, Computer Science Department, University of Maryland, College Park, MD 20742; O. Widlund, Courant Institute of Mathematical Sciences, New York University, New York, NY 10012.

scheme is known as the Shortley-Weller method. Thus the discrete system of equations has a matrix that differs from that for a Helmholtz problem on the cube only in those rows that correspond to points near Γ. We take advantage of this by reformulating the problem as one of dimension equal to the number of boundary neighbors rather than the number of mesh points in the region. The resulting linear system is the capacitance matrix equation. In our implementation this reduced equation is then solved using an iterative algorithm, the conjugate gradient method. A special scaling method, based on so-called discrete dipoles, is used to enhance the convergence. A fast Poisson solver on the cube is one of the components necessary to evaluate the product of the capacitance matrix with a given vector.

Let NX, NY, and NZ be the numbers of mesh points in the cube along the three coordinate axes, where NX and NY are powers of 2. We denote the number of mesh points in Ω with at least one of their six nearest neighbors on or outside the boundary Γ by IPP1 + IPP2, where IPP2 points have two such neighbors along at least one coordinate mesh line. The program requires as input certain scalar parameters, the coordinates of each of the IPP1 + IPP2 points and their distances to the boundary along mesh lines, the values of the Dirichlet data f at points of intersection of mesh lines and the boundary Γ, and the value of the function g at mesh points in Ω. The user communicates with the package through the subroutine HELM3D. A complete description of the input parameters is given in the comments at the beginning of this subroutine.

HELM3D controls the conjugate gradient iteration and calls upon a fast solver (CUBE) and subroutines UTAMLT, UTATRN, BNDRY, VMULT, and VTRANS to perform the necessary matrix-vector products. CUBE solves the Helmholtz equation on a cube using fast discrete Fourier transform routines RFORT and FORT to reduce the systems to tridiagonal form. The resulting linear systems are solved by a Toeplitz method, and then an inverse Fourier transform is performed. RFORT and FORT were provided by Dr. W. Proskurowski, who has modified a code written by Dr. J. Cooley. HELM3D also employs an error-checking module HELMCK to check the input data. It diagnoses errors in the integer parameters, missing boundary points, and inconsistencies in the given boundary data.

The program requires two arrays of dimension NX $\times$ NY $\times$ NZ (one if $g = 0$), four integer and six real arrays of dimension IPP1 + 2 * IPP2, and one real array of dimension max(IPP1 + 2 * IPP2, NX * NZ, NY * NZ). Each conjugate gradient iteration requires time proportional to NX * NY * NZ * LOG (NX * NY), and the number of iterations will usually be small unless a value of c is used that makes the discrete Helmholtz operator almost singular. Double precision is required on machines with a short word length.

Data on timing for many sample problems are given in [1]. As an example, a discrete Laplace problem on a cube with a sphere cut out of it having 10464 mesh points and embedded in a cube of dimension 32 $\times$ 32 $\times$ 24 required 84K words of storage and 171 seconds on a CDC 6600 (FTN compiler, OPT = 2) to find a solution of the linear system with a maximum error equal to 0.55×10^{-3} of the maximum value of the solution.

A sample driver is included with the algorithm. Possible enhancements to the algorithm are discussed in [1].

REFERENCES

1. O'LEARY, D. P., AND WIDLUND, O. Capacitance matrix methods for the Helmholtz equation on general three dimensional regions. Courant Inst. Rep. COO-3077-155, New York, Oct. 1978; also *Math. Comp. 33* (1979), 849–879.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service.]

```
      SUBROUTINE HELM3D (MODE, W, GG, NXDIM, NYDIM, NZDIM, IPP1, IPP2, DELTA, NNX
     1, NNY, NNZ, NIPDIM, NAPDIM, ICOORD, INDORD, CC, NIT, EPS, S, R, P, AP, IER)
      INTEGER MODE, NXDIM, NYDIM, NZDIM, IPP1, IPP2, NNX, NNY, NNZ, NIPDIM, ICOORD
     1(3, NIPDIM), INDORD(NIPDIM), NIT, IER
      REAL W(NXDIM, NYDIM, NZDIM), GG(NXDIM, NYDIM, NZDIM), DELTA(3, NIPDIM), CC
     1, EPS, S(NIPDIM), R(NIPDIM), P(NIPDIM), AP(NAPDIM)
C
C       THIS PROGRAM WAS DEVELOPED BY DIANNE P O'LEARY AND OLOF WIDLUND.
C       THIS IS  VERSION. MD2,  OCTOBER, 1979.
C
C       THIS PROGRAM SOLVES THE DIRICHLET PROBLEM FOR THE
C       HELMHOLTZ EQUATION OVER A GENERAL BOUNDED 3 DIMENSIONAL
C       REGION IMBEDDED IN A UNIT CUBE
C
C        -W    - W    - W    + CC*W = G1    IN THE REGION
C          XX     YY     ZZ
C
C         W = F                              ON THE BOUNDARY
C
C       WHERE F AND G1 ARE GIVEN FUNCTIONS OF X, Y, AND Z, AND  CC  IS
C       A REAL CONSTANT.  THE BOUNDARY IS ARBITRARY.  THE PROGRAM
C       PROVIDES A SOLUTION OF THE WELL KNOWN SHORTLEY-WELLER
C       APPROXIMATION OF THE DIFFERENTIAL EQUATION.  THE MESH IS UNIFORM
C       IN EACH COORDINATE DIRECTION AND A SIMPLE SEVEN POINT FORMULA
C       IS USED FOR INTERIOR MESH POINTS.  A CAPACITANCE MATRIX
C       METHOD,  WITH DISCRETE DIPOLES,  IS USED.  THE CAPACITANCE
C       MATRIX EQUATION IS FORMULATED AS A LEAST SQUARES PROBLEM
C       AND SOLVED USING THE CONJUGATE GRADIENT METHOD.
C
C     REFERENCES:
C
C       O'LEARY AND WIDLUND, CAPACITANCE MATRIX METHODS
C       FOR THE HELMHOLTZ EQUATION ON GENERAL 3-DIMENSIONAL
C       REGIONS,  NYU-DOE REPORT  COO-3077-155, OCTOBER, 1978;
C       MATH. COMP. 33, 1979 849-880.
C
C       PROSKUROWSKI AND WIDLUND,  MATH. COMP. 30, 1976 443-468.
C       ALSO  NYU-DOE REPORT.
C
C       PROSKUROWSKI, LAWRENCE BERKELEY LAB REPORTS AND
C       "NUMERICAL SOLUTION OF HELMHOLTZ'S EQUATION BY
C       IMPLICIT CAPACITANCE MATRIX METHODS," ACM TRANS.
C       ON MATH. SOFTWARE 5, 1979 36-49.
C
C       SHIEH, MRC-WISCONSIN REPORTS AND NUMER. MATH. 29
C       1978 307-327.
C
C
C     MACHINE DEPENDENT FEATURES:
C
C       THIS PROGRAM SHOULD BE CONVERTED TO DOUBLE PRECISION
C       IF IT IS TO BE USED ON COMPUTERS WITH SHORT WORD
C       LENGTH , SUCH AS IBM 360/370.
C
C
C     GENERAL DESCRIPTION OF THE PARAMETERS:
C
C       INTEGER VALUES:
C         DIMENSIONS OF ARRAYS  (NXDIM, NYDIM, NZDIM,
C             NIPDIM, NAPDIM)
C         NUMBER OF MESH POINTS IN CUBE  (NNX X NNY X
C             NNZ)
C         NUMBER OF POINTS IN REGION ADJACENT TO BOUNDARY
C             (IPP1, IPP2)
C         MAXIMUM NUMBER OF ITERATIONS ALLOWED  (NIT)
C         ERROR CODE  (IER)
C         CODE TO CONTROL PROGRAM OPTIONS  (MODE)
C
C       REAL VALUES:
C         HELMHOLTZ CONSTANT  (CC)
C         CONVERGENCE TOLERANCE  (EPS)
```

```
C
C       INTEGER ARRAYS:
C         COORDINATES OF POINTS IN REGION ADJACENT TO
C              BOUNDARY ('IRREGULAR POINTS')  (ICOORD)
C         WORK SPACE  (INDORD)
C
C       REAL ARRAYS:
C         G1  VALUES  (GG)
C         BOUNDARY VALUES  (R, P, AP)
C         DISTANCES FROM IRREGULAR POINTS TO BOUNDARY
C              (DELTA)
C         WORK SPACE  (W, S)
C
C
C       TOTAL ARRAY SPACE NEEDED:
C         REAL:   2  NXDIM * NYDIM * NZDIM  (1  IF  G1=0)
C                 6  NIPDIM
C                 1  NAPDIM
C         INTEGER:
C                 4  NIPDIM
C
C         WHERE  NXDIM .GE. NNX,  NYDIM .GE. NNY,
C                NZDIM .GE. NNZ,
C                NIPDIM .GE. IPP1 + 2 * IPP2,
C                NAPDIM .GE. MAX (IPP1+2*IPP2, NNX*NNZ,
C                                 NNY*NNZ)
C
C
C     NOTE:
C
C       IN THIS DOCUMENTATION,  NN  REFERS TO  NNX,  NNY,  OR  NNZ
C       AS APPROPRIATE, AND SIMILARLY  H REFERS TO  HX,  HY,  OR  HZ.
C       THE MESH POINT (X,Y,Z) IS SAID TO HAVE 6 NEIGHBORS:
C          (X+HX,Y,Z), (X-HX,Y,Z), (X,Y+HY,Z), (X,Y-HY,Z),
C          (X,Y,Z+HZ), AND (X,Y,Z-HZ).
C       A MESH POINT IS CALLED IRREGULAR IF IT IS IN THE INTERIOR OF
C       THE REGION AND AT LEAST ONE OF ITS SIX NEIGHBORS IS ON OR
C       OUTSIDE THE BOUNDARY.
C
C   ON INPUT . . .
C    -- MODE =  1 IF THE REGION HAS BEEN CHANGED FROM THE PREVIOUS CALL
C                 AND  G1=0
C              2 IF THE REGION HAS BEEN CHANGED FROM THE PREVIOUS CALL
C                 AND  G1  IS NONZERO
C              3 IF THE REGION IS THE SAME AS ON THE PREVIOUS CALL
C                 AND  G1=0
C              4 IF THE REGION IS THE SAME AS ON THE PREVIOUS CALL
C                 AND  G1  IS NONZERO
C              5 IF THE PROBLEM IS THE SAME AS ON THE  PREVIOUS CALL,
C                 G1=0,  AND THE ONLY CHANGE IS THAT  EPS  AND/OR  NIT
C                 MAY HAVE BEEN CHANGED
C              6 IF THE PROBLEM IS THE SAME AS ON THE PREVIOUS CALL,
C                 G1  IS NONZERO, AND THE ONLY CHANGE IS THAT  EPS
C                 AND/OR  NIT  MAY HAVE BEEN CHANGED
C        IF  MODE = 3,4,5, OR 6  DELTA,  ICOORD,  INDORD,  NXDIM,
C       NYDIM,  NZDIM,  NNX,  NNY,  NNZ,  IPP1,  AND IPP2  MUST BE
C       UNCHANGED FROM THE PREVIOUS CALL.  THE CURRENT VALUE OF  S
C       WILL BE USED AS THE INITIAL GUESS FOR THE DIPOLE STRENGTHS.
C       (S=0 WILL BE USED IF MODE=1 OR 2.)
C       TO IMPROVE THE ACCURACY OF A PREVIOUSLY CALCULATED SOLUTION,
C       USE  MODE=5  OR MODE=6  IF ROUNDOFF IS NOT SUSPECTED.  IF
C       ROUNDOFF IS SUSPECTED, REINITIALIZE THE BOUNDARY VALUES IN  R,
C       AP,  AND  P,  AND USE  MODE = 3 TO FORCE THE RESIDUAL TO BE
C       RECOMPUTED; IF G1 IS NONZERO, ADD GG TO THE SOLUTION
C       RETURNED BY THE SUBROUTINE.
C
C    -- W(NXDIM, NYDIM, NZDIM) IS UNINITIALIZED.
C    -- GG(NXDIM, NYDIM, NZDIM) INITIALIZED TO  G1*HZ*HZ  IN THE
C                               REGION, WITH ARBITRARY VALUES OUTSIDE.
C         FOR  I=1,...,NNX,  J=1,...,NNY, AND  K=1,...,NNZ,
C         GG(I,J,K) CORRESPONDS TO G1((I-1)*HX,(J-1)*HY,(K-1)*HZ)*HZ**2.
C         IF  MODE = 1, 3 OR 5,  G1  MAY BE A DUMMY ARRAY (I.E.,
C         IT NEED NOT BE DIMENSIONED BY THE CALLING PROGRAM).
C
C    -- IPP1 IS THE NUMBER OF IRREGULAR POINTS WITH AT LEAST 1
C          INTERIOR NEIGHBOR IN EACH DIRECTION X, Y, AND Z.
```

```
C     -- IPP2 IS THE NUMBER OF IRREGULAR POINTS WHICH, ALONG
C           AT LEAST ONE DIRECTION, HAVE TWO EXTERIOR NEIGHBORS.
C
C        IN THE EXCEPTIONAL CASE WHEN  IPP1+IPP2.EQ.0,  THE ROUTINE
C        WILL SOLVE THE PROBLEM ON THE WHOLE CUBE WITH THE
C        BOUNDARY CONDITIONS#
C              G1(X,Y,Z) = 0     Z .LT. 0     OR Z .GT. 1
C              W(0,Y,Z) = W(1,Y,Z)  AND  W(X,0,Z) = W(X,1,Z)
C              W(X,Y,0)=0 AND W(X,Y,Z) BOUNDED FOR ALL Z.
C        ARRAY  GG  MUST BE INITIALIZED TO  G1*HZ*HZ  AND  MODE = 2.
C        W  MAY BE A DUMMY ARRAY.  THE ANSWER WILL BE STORED
C        IN THE ARRAY  GG  IN THIS CASE.
C
C     -- DELTA(3, NIPDIM)  RECORDS + OR - DISTANCE TO BOUNDARY
C           FROM IRREGULAR POINT  L  IN THE  X,  Y,  AND  Z
C           DIRECTIONS  (3*IPP1 + 6*IPP2  VALUES).  THESE DISTANCES
C           ARE EXPRESSED AS MULTIPLES OF THE MESH SPACING:  I.E.,
C           IF A  DELTA  HAS THE VALUE  Q,  THE DISTANCE IS  Q*H.
C         THERE ARE THREE DELTAS FOR EACH OF THE  IPP1  POINTS
C         FOR  L=1,IPP1,
C          DELTA(1,L) = SHORTER DISTANCE TO BOUNDARY ALONG X DIRECTION
C          DELTA(2,L) = SHORTER DISTANCE TO BOUNDARY ALONG Y DIRECTION
C          DELTA(3,L) = SHORTER DISTANCE TO BOUNDARY ALONG Z DIRECTION
C         THERE ARE SIX DELTAS FOR EACH OF THE IPP2 POINTS,
C         FOR  L=1,IPP2  ,  LL=IPP1+2*L-1,
C          DELTA(1,LL)  AND  DELTA(1,LL+1)  ARE THE DISTANCES TO THE
C             BOUNDARY ALONG THE POSITIVE AND NEGATIVE  X  DIRECTIONS
C          DELTA(2,LL)  AND  DELTA(2,LL+1)  ARE THE DISTANCES TO THE
C             BOUNDARY ALONG THE POSITIVE AND NEGATIVE  Y  DIRECTIONS
C          DELTA(3,LL)  AND  DELTA(3,LL+1)  ARE THE DISTANCES TO THE
C             BOUNDARY ALONG THE POSITIVE AND NEGATIVE  Z  DIRECTIONS
C         THE PROGRAM WILL INTERCHANGE  DELTAS  IF NECESSARY SO THAT
C         FOR  L=1,IPP2  ,  LL=IPP1+2*L-1,
C         ABS(DELTA(S,LL))  LE.  ABS(DELTA(S,LL+1)).
C         NO  DELTA  CAN BE SO CLOSE TO  0  AS TO CAUSE OVERFLOW
C         UPON DIVISION BY A PRODUCT OF TWO  DELTAS.  SUCH SMALL
C         DELTAS  SHOULD BE AVOIDED BY CHANGING THE REGION
C         SLIGHTLY OR BY SHIFTING IT INSIDE THE CUBE OR BY
C         USING ANOTHER MESH SIZE.
C
C     -- NNX, NNY, NNZ ARE THE NUMBER OF MESH POINTS IN THE X, Y, AND Z
C                 DIRECTIONS.
C        MAX(NNX,NNY) MUST BE .LE. 256 UNLESS THE ERROR CHECK IN
C        HELMCK AND THE DIMENSIONS OF IB AND S IN COMMON FFT
C        (SUBROUTINES CUBE,RFORT AND FORT) ARE CHANGED.
C         THE MESH SPACINGS WILL BE CALCULATED TO BE
C                  HX = 1 / NNX
C                  HY = 1 / NNY
C                  HZ = 1 / (NNZ - 1)
C         NNX  AND  NNY  MUST BE POWERS OF  2  AND  .GE. 8  UNLESS
C         THE FFT ROUTINES  RFORT  AND  FORT  ARE REPLACED.
C
C     -- NIPDIM, THE DIMENSION OF THE ONE DIMENSIONAL ARRAYS,
C              MUST BE .GE. IPP1+2*IPP2.
C     -- NAPDIM , THE DIMENSION OF  AP , MUST
C              BE .GE.  MAX(IPP1+2*IPP2, NNX*NNZ, NNY*NNZ ).
C     -- ICOORD(3,NIPDIM)  RECORDS THE  3*(IPP1+IPP2)  INDICES OF
C              THE IRREGULAR POINTS.  THESE INDICES MUST LIE BETWEEN
C              2 AND NN-1 INCLUSIVE.
C              FOR  L = 1, IPP1
C                 THE  L-TH  COLUMN OF  ICOORD
C                 GIVES THE INDICES CORRESPONDING
C                 TO DATA IN THE  L-TH  COLUMNS OF
C                 DELTA,  P,  R,  AND  AP.
C              FOR  L = 1, IPP2 ,   LL = IPP1 + 2 * L - 1
C                 THE  (IPP1+L)-TH  COLUMN OF  ICOORD
C                 GIVES THE INDICES CORRESPONDING TO
C                 DATA IN THE  LL-TH  AND  (LL+1)-TH
C                 COLUMNS OF  DELTA,  P,  R,  AND  AP.
C     -- INDORD (NIPDIM)  IS UNINITIALIZED.  THE PROGRAM WILL
C              RECORD A CODE (1-6) FOR THE ORDER OF THE DELTAS.
C     -- CC  IS THE CONSTANT IN THE HELMHOLTZ EQUATION.
C     -- NIT IS THE MAXIMUM NUMBER OF CONJUGATE GRADIENT ITERATIONS
C              ALLOWED.
C     -- EPS IS THE TOLERANCE FOR THE EUCLIDEAN NORM OF
C              THE CAPACITANCE EQUATION RESIDUAL DIVIDED BY THE
```

```
C               SQRT OF THE DIMENSION OF THIS VECTOR.
C                                T T     T               T
C               RESIDUAL = C U F - C C S  WHERE C = U AGV .
C               IT IS DIFFICULT TO GIVE A RELIABLE RULE OF
C               THUMB FOR THE CHOICE OF EPS.  FOR MANY PROBLEMS
C               ONE TENTH OF THE DESIRED ACCURACY FOR THE
C               SOLUTION OF THE ORIGINAL DISCRETE PROBLEM IS A
C               SUITABLE VALUE. A SMALLER TOLERANCE IS REQUIRED
C               WHEN THE DISCRETE HELMHOLTZ OPERATOR IS CLOSE
C               TO SINGULAR.
C     -- S , P , R  ARE OF DIMENSION  NIPDIM .
C           AP  IS OF DIMENSION  NAPDIM
C               S IS UNINITIALIZED IF  MODE = 1 OR 2.
C               IF MODE .LT. 5 , FOR  L=1,IPP1+2*IPP2,
C                R(L) = F(X+DELTA(1,L)*HX,  Y,  Z)
C                P(L) = F(X,  Y+DELTA(2,L)*HY,  Z)
C                AP(L) = F(X,  Y,  Z+DELTA(3,L)*HZ)
C        WHERE X,Y,  AND Z ARE THE COORDINATES OF THE
C        IRREGULAR POINT  CORRESPONDING TO THE DELTAS.
C        THE VALUES OF  R , P , AND  AP  ARE NOT USED IN THE
C        COMPUTATION IF THE ABSOLUTE VALUE OF THE CORRESPONDING
C        DELTA  IS GREATER THAN 1.
C
C     -- IER IS UNINITIALIZED.  THE PROGRAM WILL RECORD AN ERROR
C        CODE (0-3).
C        THE USE OF DISCRETE DIPOLES IMPOSES A MILD RESTRICTION
C        ON THE GEOMETRY OF THE REGION.  THE THREE MESH POINTS, OBTAINED
C        BY STEPPING FROM AN IRREGULAR POINT IN THE DIRECTION OF THE
C        SMALLEST MAGNITUDE  DELTA,  FROM THIS NEW POINT IN
C        THE DIRECTION OF THE MEDIUM, AND FROM THERE IN THE DIRECTION
C        OF THE LARGEST MUST NOT BE INTERIOR POINTS OF THE REGION.
C        ASSOCIATED WITH AN IRREGULAR POINT WHICH HAS AT
C        LEAST  2  EXTERIOR NEIGHBORS IN SOME MESH
C        DIRECTION ARE TWO COLUMNS OF THE ARRAY  DELTA
C        THE  DELTA'S  RELEVANT TO THIS TEST ARE THE
C        SMALLER IN MAGNITUDE OF THE TWO POSSIBLE
C        CHOICES IN EACH COORDINATE DIRECTION.  IF THE
C        RESTRICTION IS VIOLATED, A SUBROUTINE  HELMCK  WILL RETURN AN
C        ERROR FLAG  IER = 2.  A REFINEMENT OF THE MESH OR
C        A SLIGHT SHIFT OF THE REGION IN THE UNIT CUBE MIGHT
C        RESOLVE THE PROBLEM.
C
C     ON OUTPUT . . .
C           W  WILL CONTAIN VALUES OF THE SOLUTION INSIDE THE
C                 REGION AND USELESS VALUES OUTSIDE AND ON THE
C                 BOUNDARY.
C           S  WILL RECORD DIPOLE STRENGTHS.  THIS IS THE SOLUTION
C                 VECTOR OF THE CAPACITANCE MATRIX EQUATION.
C           R  WILL BE THE RESIDUAL OF THE CAPACITANCE EQUATION.
C
C           P , AP , AND  GG  WILL BE CHANGED, AND THE  DELTAS  MAY
C           BE REORDERED AS INDICATED ABOVE.
C
C        ERROR RETURNS:
C            IER=0       NO ERROR
C               =1       ERROR IN INTEGER PARAMETER
C               =2       ERROR IN  ICOORD  OR VIOLATION OF DIPOLE
C                        RESTRICTION OR IRREGULAR POINT MISSING
C               =3       TOO MANY CONJUGATE GRADIENT ITERATIONS
C                        WITHOUT CONVERGENCE.  ANSWER DOES NOT
C                        HAVE THE REQUESTED ACCURACY.
C
C
C        AFTER EACH ITERATION, THE FOLLOWING INFORMATION IS PRINTED:
C           -- THE CONJUGATE GRADIENT PARAMETERS  ALPHA  AND  BETA.
C                 THIS INFORMATION COULD BE USED TO ESTIMATE THE
C                 CONDITION NUMBER OF THE CAPACITANCE MATRIX.
C           -- THE EUCLIDEAN NORM OF THE RESIDUAL OF THE
C              CAPACITANCE MATRIX EQUATION.
C                              T T   T               T
C                    THE RESIDUAL=C U F-C CS  WHERE C= U AGV.
C
C     THE ROLES OF THE SUBROUTINES:
C           HELM3D  CONTROLS THE CONJUGATE GRADIENT ITERATION.
C           HELMCK  CHECKS THE INPUT DATA FOR CORRECTNESS.
C           VMULT   USES THE DIPOLE STRENGTHS IN A  NIPDIM  ARRAY TO
```

```
C                SET UP THE DIPOLES IN A 3 DIMENSIONAL ARRAY.
C                THIS SUBROUTINE THUS DEFINES A LINEAR MAPPING
C                FROM A SPACE OF  1-DIMENSIONAL ARRAYS TO A SPACE
C                OF  3-DIMENSIONAL ARRAYS.
C          VTRANS  DEFINES THE TRANSPOSE OF THE MAPPING DEFINED
C                BY  VMULT.
C          UTAMLT  MAPS  3-DIMENSIONAL ARRAYS INTO  1-DIMENSIONAL
C                ARRAYS BY USING A FINITE DIFFERENCE FORMULA WHICH
C                CORRESPONDS TO A PART OF THE SHORTLEY-WELLER
C                APPROXIMATION.  THE REMAINING PART IS HANDLED BY
C                BNDRY.
C          UTATRN  DEFINES THE TRANSPOSE OF THE MAPPING DEFINED BY
C                UTAMLT.
C          BNDRY   PROCESSES THE DIRICHLET DATA AND THE VALUES OF  GJ
C                CLOSE TO THE BOUNDARY, PRODUCING  U(TRANSPOSE)F  FOR
C                USE IN THE RIGHT HAND SIDE OF THE CAPACITANCE EQUATION
C          CUBE    SOLVES THE HELMHOLTZ EQUATION OVER A CUBE USING A
C                FOURIER-TOEPLITZ ALGORITHM.
C          RFORT   IS A FAST FOURIER TRANSFORM ROUTINE DUE TO
C                W. PROSKUROWSKI WHO REVISED A CODE WRITTEN BY J. COOLEY.
C                IT IS USED BY SUBROUTINE CUBE.
C          FORT    IS A SUBROUTINE CALLED BY  RFORT.
```

573-P 1- 0

ALGORITHM 573
NL2SOL—An Adaptive Nonlinear Least-Squares Algorithm [E4]

JOHN E. DENNIS, JR.
Rice University
and
DAVID M. GAY and ROY E. WELSCH
Massachusetts Institute of Technology

Key Words and Phrases: unconstrained optimization, nonlinear least squares, nonlinear regression, quasi-Newton methods, secant methods
CR Categories: 5.14, 5.5
Language: FORTRAN

1. PURPOSE

Given a continuously differentiable function (residual vector) $R(x) = (R_1(x), R_2(x), \ldots, R_n(x))^T$ of p parameters $x = (x_1, x_2, \ldots, x_p)^T$, NL2SOL attempts to find a parameter vector x^* that minimizes the sum-of-squares function $F(x) = \sum_{i=1}^{n} R_i(x)^2$.

2. METHOD

Reference [1] explains the algorithm realized by NL2SOL in detail. The algorithm amounts to a variation on Newton's method in which part of the Hessian matrix is computed exactly and part is approximated by a secant (quasi-Newton) updating method. Once the iterates come sufficiently close to a local solution, they usually converge quite rapidly. To promote convergence from poor starting guesses, NL2SOL uses a model/trust-region technique along with an adaptive choice of the model Hessian. Consequently, the algorithm sometimes reduces to a Gauss–Newton or Levenberg–Marquardt method. On large residual problems (in which $F(x^*)$ is large), however, NL2SOL often works much better than these methods.

Received 13 September 1977; revised 18 August 1979 and 25 September 1980; accepted 8 April 1981.

Research leading to the NL2SOL package was supported in part by National Science Foundation Grants DCR75-10143, MCS76-00324, and SOC76-14311 to the National Bureau of Economic Research, Inc., and MCS79-06671 to the Massachusetts Institute of Technology, and was sponsored in part by NSF Grant MCS78-09525 and United States Army Contract DAAG29-75-C-0024 to the Mathematics Research Center at the University of Wisconsin.
Authors' addresses: J.E. Dennis, Jr., Department of Mathematical Sciences, Rice University, P.O. Box 1892, Houston, TX 77001; D.M. Gay, M.I.T./CCREMS, Room E38-278, Cambridge, MA 02139; R.E. Welsch, M.I.T./CCREMS, Room E53-383, Cambridge, MA 02139.

3. DESCRIPTION

3.1 Calling Sequence

CALL NL2SOL (N, P, X, CALCR, CALCJ, IV, V, UIPARM, URPARM, UFPARM)

Note: NL2SOL is written in American National Standard FORTRAN (1966) and the comments below assume that the calling program is also written in FORTRAN. These comments refer to the single-precision version of NL2SOL. In the double-precision version, all quantities termed REAL below are actually DOUBLE PRECISION.

N (input INTEGER) is the number of components in the residual vector R.

P (input INTEGER) is the number of parameters on which R depends.

X (I/O REAL array of length P) on input is an initial guess at the desired solution x^*. When NL2SOL returns, X contains the best parameter estimate found so far.

CALCR (input subroutine) computes the residual vector R = R(X) when invoked by

CALL CALCR(N, P, X, NF, R, UIPARM, URPARM, UFPARM)

When CALCR is called, NF is the invocation count for CALCR; it is included for possible use with CALCJ. If X is out of bounds (e.g., if R(X) would overflow), then CALCR should set NF to 0, which will cause a shorter step to be attempted. CALCR should not change N, P, or X and should be declared EXTERNAL in the calling program. R should be declared REAL R(N).

CALCJ (input subroutine) computes the Jacobian matrix J = J(X) of first partials, $J_{ij} = \partial J_i(X)/\partial x_j$, when invoked by

CALL CALCJ(N, P, X, NF, J, UIPARM, URPARM, UFPARM)

When CALCJ is called, NF is the invocation count for CALCR at the time when R(X) was evaluated. The X passed to CALCJ is usually the one passed to CALCR on either its most recent invocation or the one prior to it. Thus if CALCR saves intermediate results for use by CALCJ, then it is possible to tell from NF whether they are valid for the current X (or which copy is valid if two are kept). If J cannot be computed at X, then CALCJ should set NF to 0. CALCJ should not change N, P, or X and should be declared EXTERNAL in the calling program. J should be declared REAL J(N, P).

IV (I/O INTEGER array of length P + 60) on input contains certain values (such as limits on the number of iterations and calls on CALCR) that control the behavior of NL2SOL and on output contains various counts and other items of interest: see Sections 3.3 and 3.4. If IV(1) = 0 on input, then default values are supplied for the input components of both IV and V. The caller may supply nondefault values for selected components of IV and V by CALLing DFAULT(IV, V) and then assigning the appropriate nondefault values before calling NL2SOL.

V (I/O REAL array of length 93 + N(P + 3) + P(3P + 33)/2) on input contains certain values (such as convergence tolerances) that control the behavior of NL2SOL and on output contains various items of interest (such as F(X) and R(X)): see Sections 3.5 and 3.15.

UIPARM (INTEGER array of length determined by the caller) is passed without change to CALCR and CALCJ and may be used by them in any way that the caller may find convenient.

URPARM (REAL array of length determined by the caller), like UIPARM, is passed without change to CALCR and CALCJ.

UFPARM (subroutine), like UIPARM, is passed without change (and without having been invoked) to CALCR and CALCJ. If there is no need for such a subroutine, then on many systems it suffices to pass an arbitrary variable or constant for UFPARM. But if an actual subroutine is passed, then it must be declared EXTERNAL in the calling program.

3.2 Example

Let $n = 3, p = 2$, and

$$R(x) = \begin{bmatrix} x_1^2 + x_2^2 + x_1x_2 \\ \sin x_1 \\ \cos x_2 \end{bmatrix}.$$

(This problem is due to Madsen [3].) The following FORTRAN code minimizes $F(x) = \frac{1}{2}R(x)^TR(x)$, starting from the initial guess $(3, 1)^T$, using a single-precision version of NL2SOL.

```
INTEGER IV(62), UI(1)
REAL V(147), X(2), UR(1)
EXTERNAL MADR, MADJ
X(1) = 3.0
X(2) = 1.0
IV(1) = 0
CALL NL2SOL (3, 2, X, MADR, MADJ, IV, V, UI, UR, MADR)
STOP
END
SUBROUTINE MADR (N, P, X, NF, R, UI, UR, UF)
INTEGER N, P, NF, UI(1)
REAL X(P), R(N), UR(1)
EXTERNAL UF
R(1) = X(1)**2 + X(2)**2 + X(1)*X(2)
R(2) = SIN(X(1))
R(3) = COS(X(2))
RETURN
END
SUBROUTINE MADJ (N, P, X, NF, J, UI, UR, UF)
INTEGER N, P, NF, UI(1)
REAL X(P), J(N, P), UR(1)
EXTERNAL UF
J(1, 1) = 2.0*X(1) + X(2)
J(1, 2) = 2.0*X(2) + X(1)
J(2, 1) = COS(X(1))
J(2, 2) = 0.0
J(3, 1) = 0.0
J(3, 2) = -SIN(X(2))
RETURN
END
```

The main program above passes MADR as CALCR and MADJ as CALCJ. No use is made of UIPARM, URPARM, or UFPARM in this simple example.

When the above is executed, NL2SOL prints the initial X vector, a summary of the iterations performed, the final X vector, and some statistics, including the final F(X) and a covariance matrix. If REAL is changed to DOUBLE PRECISION and the above is run on an IBM 370 computer, then NL2SOL reports relative function convergence (IV(1) = 4—see Section 3.3) after 12 calls on MADR and MADJ and returns X(1) = −0.155437, X(2) = 0.694564, and F(X) = 0.386600.

If, say, we wanted to suppress the iteration summary, we could do so by replacing the statement IV(1) = 0 in the main program by

```
CALL DFAULT(IV, V)
IV(19) = 0
```

(See the description of IV(OUTLEV) in Section 3.4.)

3.3 Return Codes

When NL2SOL returns, IV(1) contains one of the following return codes:

3	= X-convergence. The scaled relative difference between the current parameter vector X and a locally optimal parameter x^* is very likely at most V(XCTOL): see Section 3.5.
4	= relative function-convergence. The relative difference between the current function value and its locally optimal value is very likely at most V(RFCTOL): see Section 3.5.
5	= both X and relative function-convergence, that is, the conditions for IV(1) = 3 and IV(1) = 4 both hold.
6	= absolute function-convergence. The current function value (half the sum of squares) is at most V(AFCTOL): see Section 3.5.
7	= singular convergence. The Hessian near the current X appears to be singular or nearly so, and a step of scaled length at most V(LMAX0) is unlikely to yield a relative function decrease of more than V(RFCTOL). This means that the model is overspecified (i.e., contains too many parameters), at least near X. It is possible that a different starting guess would lead NL2SOL to find an X giving a smaller F(X) and strong convergence (IV(1) = 3, 4, 5, or 6).
8	= false convergence. The iterates appear to be converging to a noncritical point. This may mean that the false convergence tolerance (V(XFTOL)—see Section 3.5) is too large, that the convergence tolerances (V(AFCTOL), V(RFCTOL), V(XCTOL)) are too small for the accuracy to which CALCR and CALCJ compute R and J, that there is an error in computing the Jacobian matrix J, or that R is discontinuous near X. If the NPRELDF value printed in the summary statistics (or in the iteration summary for the final iteration) is negative and not too much larger than V(RFCTOL) in absolute value, then V(RFCTOL) is too small and singular convergence would be detected if V(RFCTOL) were increased above |NPRELDF|: see Sections 3.5 and 3.11.
9	= function evaluation limit reached without other convergences: see IV(MXFCAL) in Section 3.4.
10	= iteration limit reached without other convergence: see IV(MXITER) in Section 3.4.
11	= STOPX returned .TRUE. (external interrupt): see Section 3.14.
13	= F(X) cannot be computed at the initial X.
14	= bad parameters passed to ASSESS. (This should not occur.)
15	= the Jacobian could not be computed at X (see CALCJ above).
16	= N or P (or parameter NN to NL2ITR) out of range: P < 0 or N < P or NN < N.
17	= a restart was attempted with N, P, or parameter NN to NL2ITR changed: see Section 3.7.
18	= IV(INITS) out of range: see Section 3.4.
19–45	= V(IV(1)) is out of range.
50	= IV(1) was out of range when NL2SOL (or NL2SNO or NL2ITR) was called.
87 . . . (86 + P)	= JTOL(IV(1)-86), that is, V(IV(1)), is not positive: see V(DFAC) in Section 3.5.

Just before NL2SOL returns, a brief description of the return code is printed (unless all printing is turned off by IV(PRUNIT) = 0).

3.4 IV Values

3.4.1 *IV Input Values (Supplied by DFAULT)*

IV(1) . . . IV(1) should have a value between 0 and 12 when NL2SOL is called. 0 and 12 both mean that this is a fresh start; 0 means DFAULT(IV, V) should be invoked to supply default values to the input components of IV and V, while 12 (the value that DFAULT assigns to IV(1)) means that the caller has already called DFAULT(IV, V) and has possibly changed some IV or V entries to nondefault values. IV(1) input values between 3 and 11 mean that NL2SOL should restart; see Section 3.7. Default = 12.

IV(COVPRT) . . . IV(14) = 1 means print a covariance matrix at the solution. This matrix is computed as IV(COVREQ) dictates just before a return with IV(1) = 3, 4, 5, or 6. IV(COVPRT) = 0 means skip this printing. Default = 1.

IV(COVREQ): IV(15) $\neq$ 0 means compute a covariance matrix before a return with IV(1) = 3, 4, 5, or 6. In this case, an approximate covariance matrix is obtained in one of several ways. Let k = | IV(COVREQ) | and let σ = 2F(X)/ max (1, N−P), where 2F(X) is the residual sum of squares. If k = 1 or 2, then a finite-difference Hessian approximation H is obtained. If H is positive-definite (or, for k = 3, if the Jacobian has full rank), then one of the following is computed:

$$k = 1 \Rightarrow \sigma H^{-1}(J^{\mathrm{T}}J)H^{-1}$$

$$k = 2 \Rightarrow \sigma H^{-1}$$

$$k = 3 \Rightarrow \sigma(J^{\mathrm{T}}J)^{-1}.$$

If IV(COVREQ) > 0, then both function and gradient values (calls on CALCR and CALCJ) are used in computing H (with step sizes determined by V(DELTA0); see Section 3.5), while if IV(COVREQ) < 0, then only function values (calls on CALCR) are used (with step sizes determined by V(DLTFDC)). If IV(COVREQ) = 0, then no attempt is made to compute a covariance matrix (unless IV(COVPRT) = 1, in which case NL2SOL assumes IV(COVREQ) = 1 and NL2SNO assumes IV(COVREQ) = −1). See IV(COVMAT) below. Default = 1.

IV(DTYPE) . . . IV(16) tells how the scale vector D (see [1]) should be chosen. IV(DTYPE) > 0 means choose D as described below with V(DFAC). IV(DTYPE) $\leq$ 0 means the caller has chosen D and has stored it in V starting at V(94 + 2N + P(3P + 31)/2). Default = 1.

IV(INITS) . . . IV(25) tells how the S matrix (see [1]) should be initialized. 0 means initialize S to all zeros and start with the Gauss–Newton model. 1 and 2 mean that the caller has stored the lower triangle of the initial S rowwise in V starting at V(87 + 2P). IV(INITS) = 1 means start with the Gauss–Newton model, while IV(INITS) = 2 means start with the augmented model; see [1]. Default = 0.

IV(MXFCAL) . . . IV(17) gives the maximum number of function evaluations (calls on CALCR, excluding those used to compute the covariance matrix and, in the case of NL2SNO, the Jacobian matrices) allowed. If this number does not suffice, then NL2SOL returns with IV(1) = 9. Default = 200.

IV(MXITER) . . . IV(18) gives the maximum number of iterations allowed. It also indirectly limits the number of gradient evaluations (calls on CALCJ) to IV(MXITER) + 1. If IV(MXITER) iterations do not suffice, then NL2SOL returns with IV(1) = 10. Default = 150.

IV(OUTLEV) . . . IV(19) controls the number and length of iteration summary lines printed (by ITSMRY). IV(OUTLEV) = 0 means do not print any summary lines. Otherwise print a summary line after each | IV(OUTLEV) |

iterations. Long summary lines are printed if IV(OUTLEV) > 0, short lines if IV(OUTLEV) < 0. See Section 3.11 for more details. Default = 1.

IV(PARPRT) ... IV(20) = 1 means print any nondefault V values on a fresh start or any changed V values on a restart. IV(PARPRT) = 0 means skip this printing. Default = 1.

IV(PRUNIT) ... IV(21) is the output unit number on which all printing is done. IV(PRUNIT) = 0 means suppress all printing. (Setting IV(PRUNIT) to 0 is the only way to suppress the one-line termination message printed before NL2SOL returns.) Default = standard output unit (unit 6 on most systems); the default for IV(PRUNIT) is actually IMDCON(1); see Section 3.12.

IV(SOLPRT) ... IV(22) = 1 means print the final X (the one returned), along with the final gradient and scale vector D. IV(SOLPRT) = 0 means skip this printing. Default = 1.

IV(STATPR) ... IV(23) = 1 means print summary statistics upon returning. These consist of the function value (half the residual sum of squares) at X, the scaled relative size of the last step taken (see V(RELDX) below), the number of function and gradient evaluations (calls on CALCR and CALCJ, excluding any calls made only for computing covariance matrices), the relative function reductions predicted for the last step taken and for a Newton step (or perhaps a step of length bounded by V(LMAX0)—see the descriptions of PRELDF and NPRELDF in Section 3.11 below), and, if an attempt was made to compute a covariance matrix, the number of calls on CALCR and CALCJ used in trying to compute the covariance matrix. IV(STATPR) = 0 means skip this printing. Default = 1.

IV(X0PRT) ... IV(24) = 1 means print the initial X and scale vector D (on a fresh start only). IV(X0PRT) = 0 means skip this printing. Default = 1.

3.4.2 *IV Output Values of Primary Interest*

IV(1) ... IV(1) is the return code; see Section 3.3.

IV(COVMAT) ... IV(26) tells whether a covariance matrix was computed. If IV(COVMAT) > 0, then the lower triangle of the covariance matrix is stored row-wise in V, starting at V(IV(COVMAT)). If IV(COVMAT) = 0, then no attempt was made to compute a covariance matrix. If IV(COVMAT) = −1, then the finite-difference Hessian H was indefinite (or, for | IV(COVREQ) | = 3, the current Jacobian matrix is rank deficient): like singular convergence (see Section 3.3), this may mean that the model is overspecified (contains too many parameters). And if IV(COVMAT) = −2, then a successful finite-difference step could not be found for some component of X (i.e., CALCR set NF to 0 for each of two trial steps).

Note that IV(COVMAT) is reset to 0 after each successful step, so if such a step is taken after a restart, then the covariance matrix will be recomputed.

IV(D) ... IV(27) is the starting subscript in V of the current scale vector D.

IV(G) ... IV(28) is the starting subscript in V of the current least-squares gradient vector $J^T R$.

IV(NFCALL) ... IV(6) is the number of calls so far made on CALCR (i.e., function evaluations, including those used in computing covariance matrices).

IV(NFCOV) ... IV(40) is the number of calls made on CALCR when computing covariance matrices.

IV(NGCALL) ... IV(30) is the number of calls on CALCJ (gradient evaluations) so far made, including those used in computing covariance matrices.

IV(NGCOV) . . . IV(41) is the number of calls made on CALCJ when computing covariance matrices.

IV(NITER) . . . IV(31) is the number of iterations performed.

IV(R) . . . IV(50) is the starting subscript in V of the residual vector R corresponding to the final X.

3.5 V Values of Primary Interest

Many of the V input components described here and in Section 3.15 must lie in certain intervals. If such a component lies outside the interval indicated for it below (or in Section 3.15) at the beginning of its description, then module PARCHK will print an error message (unless IV(PRUNIT) = 0) and will force NL2SOL to return immediately with IV(1) > 18.

Frequent reference is made below to two quantities: MACHEP and the scale vector D. MACHEP is the unit roundoff for the floating-point arithmetic being used—see Section 3.12. The scale vector D is the diagonal of the diagonal scale matrix D_k discussed in [1, Sections 5 and 7]; this scale matrix is denoted by diag(D) below.

3.5.1 *V Input Values of Primary Interest (Supplied by DFAULT)*

V(AFCTOL) . . . V(31) > 0 is the absolute function convergence tolerance. If NL2SOL finds a point where the function value (half the sum of squares) is less than V(AFCTOL), and if NL2SOL does not return with IV(1) = 3, 4, or 5, then it returns with IV(1) = 6.

$$\text{Default} = \max\{10^{-20}, \text{MACHEP}^2\}.$$

V(DELTA0) . . . V(44) $\in$ [MACHEP, 1] is a factor used in choosing the finite-difference step sizes used in computing covariance matrices when IV(COVREQ) = 1 or 2. For differences involving X(i), step size

$$\text{V(DELTA0)} \cdot \max\{|\text{X}(i)|, 1/\text{D}(i)\} \cdot \text{sign}(\text{X}(i))$$

is used, where D is the current scale vector; see [1]. If this results in CALCR setting NF to 0, then −0.5 times this step is also tried. Default = MACHEP$^{1/2}$.

V(DFAC) . . . V(41) $\in$ [0, 1] and the D0 and JTOL arrays (see V(D0INIT) and V(JTINIT)) are used in updating the scale vector D when IV(DTYPE) > 0. (D is initialized according to V(DINIT).) Let

$$\text{D1}(i) = \max\{[\text{JCNORM}(i)^2 + \max\{S_{ii}, 0\}]^{1/2}, \text{V(DFAC)D}(i)\},$$

where JCNORM(i) is the 2-norm of the ith column of the current Jacobian matrix and S is the S matrix of [1]. If IV(DTYPE) = 1, then D(i) is set to D1(i) unless D1(i) < JTOL(i), in which case D(i) is set to max{(D0(i), JTOL(i)}. If IV(DTYPE) > 1, then D is updated during the first iteration as for IV(DTYPE) = 1 (after any initialization due to V(DINIT)) and is left unchanged thereafter. Default = 0.6.

V(DINIT) . . . V(38) ≥ -10: if V(DINIT) ≥ 0, then it is the value to which all components of the scale vector D are initialized during a fresh start. Default = 0.

V(DLTFDC) . . . V(40) $\in$ [MACHEP, 1] helps choose the step sizes used in computing covariance matrices when IV(COVREQ) = −1 or −2. For differences involving X(i), the step size first tried is

$$\text{V(DLTFDC)} \cdot \max\{|\text{X}(i)|, 1/\text{D}(i)\},$$

where D is the current scale vector (see [1]). If this step is too big the first time it is tried, that is, if CALCR sets NF to 0, then −0.5 times this step is also tried. Default = MACHEP$^{1/3}$.

V(DLTFDJ) ... V(36) ∈ [MACHEP, 1] helps choose the step sizes used when NL2SNO computes a finite-difference approximation to the Jacobian matrix. For differences involving X(i), the step size first tried is

$$\mathrm{V(DLTFDJ)} \cdot \max\{|\,\mathrm{X}(i)\,|, 1/\mathrm{D}(i)\},$$

where D is the current scale vector (see [1]). If the first step is too big, that is, if CALCR sets NF to 0, then smaller steps are tried until the step size is shrunk below 1000 · MACHEP. Default = $\mathrm{MACHEP}^{1/2}$.

V(D0INIT) ... V(37) ≥ 0: if V(D0INIT) > 0, it is the value to which all components of the D0 vector (see V(DFAC)) are initialized. If V(D0INIT) = 0, then it is assumed that the caller has stored D0 in V starting at V(P + 87). Default = 1.0.

V(JTINIT) ... V(39) ≥ 0: if V(JTINIT) > 0, it is the value to which all components of the JTOL array (see V(DFAC)) are initialized. If V(JTINIT) = 0, then it is assumed that the caller has stored JTOL in V starting at V(87). Default = 10^{-6}.

V(LMAX0) ... V(35) > 0 gives the maximum 2-norm allowed for diag(D) times the very first step that NL2SOL attempts. It is also used in testing for singular convergence: if the function reduction predicted for a step of length bounded by V(LMAX0) is at most V(RFCTOL) $|F_0|$, where F_0 is the function value at the start of the current iteration, and if NL2SOL does not return with IV(1) = 3, 4, 5, or 6, then it returns with IV(1) = 7. Default = 100.

V(RFCTOL) ... V(32) ∈ [MACHEP, 0.1] is the relative function-convergence tolerance. If the current model predicts a maximum possible function reduction (see V(NREDUC)) of at most V(RFCTOL) $|F_0|$, where F_0 is the function value at the start of the current iteration, and if the last step attempted achieved no more than twice the predicted function decrease, then NL2SOL returns with IV(1) = 4 (or 5). Default = $\max\{10^{-10}, \mathrm{MACHEP}^{2/3}\}$.

V(TUNER1) ... V(26) ∈ [0, 0.5] helps decide when to check for false convergence and to consider switching models. This is done if the actual function decrease from the current step is no more than V(TUNER1) times its predicted value. Default = 0.1.

V(XCTOL) ... V(33) ∈ [0, 1] is the X-convergence tolerance. If a Newton step (see V(NREDUC)) is tried that has V(RELDX) ≤ V(XCTOL) and if this step yields at most twice the predicted function decrease, then NL2SOL returns with IV(1) = 3 (or 5). Default = $\mathrm{MACHEP}^{1/2}$.

V(XFTOL) ... V(34) ∈ [0, 1] is the false-convergence tolerance. If a step is tried that gives no more than V(TUNER1) times the predicted function reduction and that has V(RELDX) ≤ V(XFTOL), and if NL2SOL does not return with IV(1) = 3, 4, 5, 6, or 7, then it returns with IV(1) = 8. (See the description of V(RELDX) below.) Default = 100·MACHEP.

V(*) ... DFAULT supplies to V a number of tuning constants, with which it should normally be unnecessary to tinker. See Section 3.15.

3.5.2 *V Output Values of Primary Interest*

V(DGNORM) ... V(1) = $\| \mathrm{diag}(D)^{-1} g \|_2$, where g is the most recently computed gradient and D is the corresponding scale vector.

V(DSTNRM) ... V(2) = $\| \mathrm{diag}(D) \Delta x \|_2$, where Δx is the most recently computed step and D is the current scale vector.

V(F) ... V(10) is the current function value (half the residual sum of squares).

V(F0) ... V(13) is the function value at the start of the current iteration.

V(NREDUC) ... V(6), if positive, is the maximum function reduction possible according to the current model, that is, the function reduction predicted for a Newton step: $\Delta x = -H^{-1}g$, where $g = J^{T}R$ is the current gradient and H is the current Hessian approximation:

$H = J^{T}J$ for the Gauss–Newton model

$H = J^{T}J + S$ for the augmented model.

V(NREDUC) = 0 means H is not positive definite.

If V(NREDUC) < 0, then V(NREDUC) is used in the singular convergence test: It is the negative of the function reduction predicted for a step computed with a step bound of V(LMAX0).

V(PREDUC) ... V(7) is the function reduction predicted (by the current quadratic model) for the current step. This (divided by V(F0)) is used in testing for relative function convergence.

V(RADIUS) ... V(8) is the trust region radius (i.e., step bound) used for the last step tried.

V(RELDX) ... V(17) is the scaled relative change in X caused by the current step, Δx, computed as

$$\max_i\{|D(i)[X(i) - X_0(i)]|\}/\max_i\{D(i)[|X(i)| + |X_0(i)|]\},$$

where $X = X_0 + \Delta x$.

3.6 Finite-Difference Jacobians: NL2SNO

Those who do not wish to code a subroutine CALCJ for (analytically) computing the Jacobian matrix may avoid doing so by calling NL2SNO instead of NL2SOL. NL2SNO computes an approximate Jacobian matrix by forward differences (using a step size determined by V(DLTFDJ)—see Section 3.5). The calling sequence for NL2SNO amounts to the one for NL2SOL with CALCJ omitted:

```
CALL NL2SNO (N, P, X, CALCR, IV, V, UIPARM, URPARM, UFPARM)
```

The parameters for NL2SNO are the same as the corresponding ones for NL2SOL, with the minor exception of IV(COVREQ): If IV(COVPRT) = 1 and IV(COVREQ) = 0, then NL2SNO sets IV(COVREQ) to −1; otherwise, it sets IV(COVREQ) to − | IV(COVREQ) | . Thus NL2SNO uses function values only in computing covariance matrices and V(DELTA0) is not used.

3.7 Restarting

After any return with $3 \le$ IV(1) ≤ 11, it is possible to change some of the IV and V input components (such as the convergence tolerances and the iteration and function evaluation limits) and call NL2SOL (or NL2SNO) again with IV(1) unchanged. This causes the algorithm to be resumed at the point where it was interrupted. (It is even possible to save IV, V, and X and then restart in a subsequent run.)

3.8 Scaling

Problems sometimes arise that are poorly scaled in the sense that the various components of X are expressed in widely differing units. With the default choice of the scale vector D (see V(DFAC) and the beginning of Section 3.5), the behavior of NL2SOL is largely insensitive to this kind of poor scaling. On well-scaled problems, the performance of NL2SOL can sometimes be improved by choosing D to be a vector of ones, that is, by setting IV(DTYPE) to 0 and V(DINIT) to 1.0. Occasionally it may also be worthwhile to fix D(i), $1 \le i \le$ P, at the 2-norm of the ith column of the initial Jacobian matrix by setting IV(DTYPE) to 2.

3.9 LMAX0: The Initial Step Bound

On some problems it is necessary to give V(LMAX0) = V(35) a small value to prevent a disasterously large first step, one that might result in exponent overflow or arguments out of range to intrinsic functions. Even if no disaster occurs, if NL2SOL takes a number of function evaluations on the first iteration, then this number can be reduced on subsequent reruns by setting V(LMAX0) to the value in the D*STEP column of the iteration summary for iteration 1.

3.10 Local Solutions

It can easily happen that NL2SOL only finds a local minimizer of the sum-of-squares function F(X) and that a different starting guess would cause a point to be found at which F has a still smaller value. Except for cases where special conditions (such as convexity of the objective function) prevail, this shortcoming is shared by all minimization algorithm implementations.

3.11 Printed Output

Any printing is done by one of two modules: ITSMRY and PARCHK. PARCHK reports any V input components that are out of range and optionally lists any such components that have nondefault or changed values (on a fresh start or restart, respectively). ITSMRY does the remaining printing. Various IV input components control what printing is done; see Section 3.4.

If IV(OUTLEV) > 0, then ITSMRY produces an iteration summary which includes the following values: IT, the current iteration number; NF, the number of function evaluations (calls on CALCR), excluding any extra ones needed for computing covariance matrices and, in the case of NL2SNO, excluding the extra ones needed to compute finite-difference Jacobian matrices; F, the current function value (half the residual sum of squares); RELDF, the relative difference between the previous and the current function value (i.e., the difference in function values divided by the previous function value); PRELDF, the value of RELDF predicted by the quadratic model used to compute the step just taken; RELDX, the relative change in X caused by the step just taken—see V(RELDX) in Section 3.5; MODEL, a code that tells which models were used in choosing the current step (G = the Gauss–Newton model; S = the augmented model; G-S means the Gauss–Newton model was tried first and a switch was then made to the augmented model; S-G, G-S-G, and S-G-S have analogous meanings); STPPAR, the Marquardt parameter λ for the last step, Δx, computed: $\lambda > 0$ means Δx satisfies

$$[H + \lambda \operatorname{diag}(D)^2]\Delta x = -g,$$

where H and g are the old Hessian approximation and gradient, respectively (the ones used in computing the step just taken); SIZE, the sizing factor used in updating the S matrix (see [1]); D*STEP, the 2-norm of diag(D) times the step just taken (see V(DSTNRM) in Section 3.5); and NPRELDF: if NPRELDF > 0, then it is the relative function reduction (i.e., value of RELDF) predicted for a full Newton step; if NPRELDF = 0, then the Hessian approximation failed to be positive definite; and if NPRELDF < 0, then it is the negative of the relative function reduction predicted for a step of length bounded by V(LMAX0). These summary lines are produced every IV(OUTLEV) iterations, and they are 118 characters long (including the carriage control character). If IV(OUTLEV) < 0, then short summary lines are produced every −IV(OUTLEV) iterations; these lines are 79 characters long (55 if IV(COVPRT) = 0), and they include only the first six items listed above (i.e., IT, NF, F, RELDF, PRELDF, and RELDX).

3.12 Changing Computers

The NL2SOL distribution tape contains both single- and double-precision versions of the NL2SOL source code, so it should be unnecessary to change precisions. (On computers having only 32 or 36 bits per REAL word, double precision often gives better performance.)

Only the functions IMDCON and RMDCON contain machine-dependent constants. To change from one computer to another, it should suffice to change the DATA statements in these functions. The DATA statement in IMDCON sets IMDCON(1) to the output unit number that DFAULT supplies to IV(PRUNIT). The machine-dependent DATA statement in RMDCON provides three values: BIG, ETA, and MACHEP. BIG is the largest floating-point number such that a FORTRAN program can compute SQRT(0.999*BIG)**2 (i.e., DSQRT(0.999D0*BIG)**2 in double precision) without overflowing. Similarly, ETA is the smallest floating-point number such that SQRT(1.001*ETA)**2 (or DSQRT(1.001D0*ETA)**2, respectively) does not underflow. MACHEP is the unit roundoff, that is, the smallest floating-point number such that 1 + MACHEP yields a stored floating-point number greater than 1. (Some computers feature registers that carry more bits than can be stored; MACHEP should only reflect the accuracy of numbers that can be stored.) DATA statements giving suitable values for BIG, ETA, and MACHEP for a variety of computers appear as comments in RMDCON.

Intrinsic functions are explicitly declared in the NL2SOL source code. On certain computers (e.g., Univac), it may be necessary to comment out these declarations. So that this may be done automatically by a simple program, such declarations are preceded by a comment having C/+ in columns 1–3 and blanks in columns 4–72 and are followed by a comment having C/ in columns 1 and 2 and blanks in columns 3–72.

3.13 Using Reverse Communication: NL2ITR

Instead of writing subroutines CALCR and CALCJ to compute the residual vector R(X) and Jacobian matrix J(X), one can call NL2ITR and provide R and J by reverse communication. The calling sequence is

```
CALL NL2ITR (D, IV, J, N, NN, P, R, V, X)
```

Parameters IV, N, P, V, and X are the same as the corresponding ones to NL2SOL, with the following exceptions: V need only contain 93 + 2N + P(3P + 31)/2 elements, since the storage that NL2SOL and NL2SNO allocate for D, J, and R at the end of V is not needed; and components IV(D), IV(J), and IV(R) are not referenced. D is the scale vector (dimensioned D(P)). NN is the (integer) lead dimension for the J array, which is dimensioned J(NN, P); NN must satisfy NN $\geq$ N.

When NL2ITR is first called (with IV(1) = 0 or 12), J must have been set to J(X), R to R(X). When NL2ITR wants R to be evaluated at a new X, it returns with IV(1) = 1; the caller should then set R to R(X) (unless X is out of range, in which case the caller should set IV(TOOBIG), that is, IV(2), to 1) and call NL2ITR again. Similarly, when NL2ITR wants J to be evaluated at X, it returns with IV(1) = 2, and the caller should then set J to J(X) and call NL2ITR again. (If J cannot be evaluated at X, the caller may set IV(NFGCAL), that is, IV(7), to 0; this will cause NL2ITR to give the error return IV(1) = 15.)

3.14 STOPX

It is possible to arrange for NL2SOL (NL2SNO and NL2ITR) to be interrupted before each evaluation of R(X) when used in an interactive environment. To do this, it is necessary to replace the logical function STOPX supplied with the NL2SOL package (which always returns .FALSE.) by a system-dependent STOPX that returns .TRUE. if and only if the "break" (i.e., "interrupt") key has been pressed since the last call on STOPX. Once this is done, NL2SOL will return with IV(1) = 11 when the "break" key is pressed before some other return has occurred. It is then possible to change some of the IV and V input components and restart; see Section 3.7.

3.15 Other V Input Values

V(COSMIN) ... V(43) $\in$ [MACHEP, 1] is the minimum absolute cosine allowed between the step just taken, Δx, and the corresponding change in gradients, Δg, for a full update of the S matrix to be made. If $|\Delta x^T \Delta g| / (\|\Delta x\|_2 \cdot \|\Delta g\|_2) <$ V(COSMIN), then $\Delta x^T \Delta g$ is replaced in the update formula by sign $(\Delta x^T \Delta g)$V(COSMIN) $\|\Delta x\|_2 \|\Delta g\|_2$. Default = max{$10^{-6}$, 100 MACHEP}.

V(DECFAC) ... V(22) $\in$ [0.01, 0.8] is the factor by which the trust region radius is shrunk if CALCR sets NF to 0 (or NL2ITR is called with IV(1) = 1 and IV(TOOBIG) = 1). Default = 0.5.

V(EPSLON) ... V(19) $\in$ [0.001, 0.9] is the maximum relative difference allowed between $g^T \Delta x + 1/2 \Delta x^T H \Delta x$ and its optimal value subject to the constraint $\|\text{diag}(D)\Delta x\|_2 \leq$ V(RADIUS), where Δx is the step being computed, g is the current gradient, and H is the current Hessian approximation. This is used in detecting and handling the special case discussed in [2]. Default = 0.1.

V(FUZZ) ... V(45) $\in$ [1.01, 100] is used in the test that decides whether to switch models. If q is the current model for F (near the point X) and $\tilde{q}$ is the other model, and if

$$\text{V(FUZZ)}\,|\tilde{q}(\text{X} + \Delta x) - \text{F}(\text{X} + \Delta x)| < |q(\text{X} + \Delta x) - \text{F}(\text{X} + \Delta x)|,$$

then the models are switched. Default = 1.5.

V(INCFAC) ... V(23) $\in$ [1.2, 100] is the minimum factor by which the trust region radius is increased (when it is increased at all). Default = 2.

V(PHMNFC) ... V(20) $\in$ [−0.99, −0.001] is the minimum value allowed for $[\|\text{diag}(D)\Delta x\|_2 - \text{V(RADIUS)}]/\text{V(RADIUS)}$. Default = −0.1.

V(PHMXFC) ... V(21) $\in$ [1.2, 100] is the maximum value allowed for $[\|\text{diag}(D)\Delta x\|_2 - \text{V(RADIUS)}]/\text{V(RADIUS)}$. Default = 0.1.

V(RDFCMN) ... V(24) $\in$ [0.01, 0.8] is the minimum factor by which the trust region radius, V(RADIUS), may be shrunk. Default = 0.1.

V(RDFCMX) ... V(25) $\in$ [1.2, 100] is the maximum factor by which the trust region radius, V(RADIUS), may be increased at one time. Default = 4.0.

V(RLIMIT) ... V(42) $\geq 10^{10}$ is the largest value allowed for $\|\text{R(X)}\|_2$ before F(X) is considered to overflow. Default = $(0.999 \cdot \text{BIG})^{1/2}$, where BIG is described in Section 3.12.

V(TUNER2) ... V(27) $\in$ [0, 0.5]. For a step to be accepted, the actual function reduction must be more than V(TUNER2) times its predicted value. Default = 0.0001.

V(TUNER3) ... V(28) $\in$ [0.001, 1]. If the actual function decrease is at least V(TUNER3) times the inner product of the step and the gradient (at the start of the step), then the trust region radius is increased. Default = 0.75.

V(TUNER4) ... V(29) $\in$ [−1, 1]. If the disposition of the new radius has not yet been decided and either

$$\|\text{diag}(D)^{-1}[H\Delta x - (g - g_0)]\| < \text{V(TUNER4)}\,\|\text{diag}(D)^{-1} g\|$$

or $g^T \Delta x <$ V(TUNER5) $g_0^T \Delta x$, where Δx is the step just taken, H is the Hessian approximation used in computing Δx, g_0 is the old gradient, g is the new gradient, and D is the newly updated scale vector, then the radius is increased by a factor of V(INCFAC). Otherwise, it is left unchanged. Default = 0.5.

V(TUNER5) ... V(30) $\geq$ MACHEP is described above with V(TUNER4). Default = 0.75.

3.16 Storage Requirements

NL2SOL, NL2SNO, and the subroutines from the NL2SOL package that they call amount to around 2360 FORTRAN statements (including nonexecutable statements, such as type statements, but excluding comments); the many comments bring the source code up to nearly 5200 lines. When compiled by the *H*-extended compiler on the IBM 370/168 at the Massachusetts Institute of Technology, this source code results in about 56,300 bytes of object code. The amount of variable storage used is listed above in Section 3.1.

REFERENCES

1. DENNIS, J.E., GAY, D.M., AND WELSCH, R.E. An adaptive nonlinear least-squares algorithm. *ACM Trans. Math. Softw. 7*, 3 (Sept. 1981), 348–368.
2. GAY, D.M. Computing optimal locally constrained steps. *SIAM J. Sci. Statist. Comput. 2*, 2 (June 1981), 186–197.
3. MADSEN, K. An algorithm for minimax solution of overdetermined systems of nonlinear equations. Rep. TP 559, AERE Harwell, England, 1973.

ALGORITHM

[The complete listing is available from the ACM Algorithms Distribution Service.]

ALGORITHM 574
Shape-Preserving Osculatory Quadratic Splines [E1, E2]

D. F. McALLISTER
North Carolina State University
and
J. A. ROULIER
University of Connecticut

Key Words and Phrases: polynomial interpolation, osculation, shape preserving, convexity preserving, monotonicity preserving, Bernstein polynomial
CR Categories: 5.18, 8.2
Language: FORTRAN

DESCRIPTION

This algorithm is a FORTRAN implementation for the procedure developed in [3]. Let n data points $\{(x_i, y_i)\}_{i=1}^n$ and n first derivatives $\{m_i\}_{i=1}^n$ at these data points be given, with $x_i \leq x_{i+1}$, $1 \leq i \leq n-1$. The algorithm constructs a smooth osculatory quadratic spline S, which satisfies

(1) $S(x_i) = y_i$, $1 \leq i \leq n$;
(2) $S'(x_i) = m_i$, $1 \leq i \leq n$;
(3) S preserves monotonicity and convexity in the case that the slopes m_i are consistent with the shape of the data;
(4) the knots of the spline S include the data points and at most two additional knots between adjacent data points.

The spline S is a piecewise quadratic Bernstein polynomial with a continuous first derivative.

Subroutine SLOPES

This subroutine will calculate values for the slopes $\{m_i\}_{i=1}^n$, which will always be consistent with the shape of the data and which guarantee that at most one additional knot is required between adjacent data points. This subroutine is called before the main routine MEVAL if the user does not wish to provide these values.

Received 26 April 1979, revised 11 August 1980, accepted 4 February 1981.

The research of D. F. McAllister was supported in part by NASA Grant NSG-1549. The research of J. A. Roulier was supported in part by NASA Grant NSG-1549, in part by U.S. Department of Energy Grant EY-71-S-09-0902, and in part by U.S. Department of The Interior, Bureau of Land Management Contract 025-08.
Authors' addresses: D. F. McAllister, Department of Computer Science, North Carolina State University, Raleigh, NC 27650; J. A. Roulier, Department of Mathematics, University of Connecticut, Storrs, CT 06268.

Subroutine MEVAL

The input to MEVAL is the set of data points $\{(x_i, y_i)\}_{i=1}^{n}$, the slopes $\{m_i\}_{i=1}^{n}$, and a vector of abscissas $u = (u_1, \ldots, u_m)$ at which the spline S is to be evaluated. In addition, a nonnegative tolerance parameter EPS must be specified for use by the subroutine CHOOSE to distinguish various cases which the algorithm must consider. This parameter is described later. If an argument u_j is determined by the subroutine SEARCH to lie in the interval $[x_i, x_{i+1}]$, then MEVAL calls the subroutines CHOOSE and CASES, in that order, to compute the parameters of S for the interval. The subroutine then calls a function subprogram SPLINE which evaluates S at the argument u_j. Extrapolation is performed for those components of u that lie outside the interval $[x_1, x_n]$.

On exit from MEVAL the error parameter ERR is zero if processing proceeded normally and no extrapolation was required. ERR is 1 if at least one component of the vector u lies outside the interval $[x_1, x_n]$. A value of 2 indicates that the components of the vector u were not in ascending order, in which case no values of S are returned.

Subroutine SEARCH

This subroutine is a binary search used by MEVAL to locate the interval in which an argument of S lies.

Subroutine CHOOSE

The value of S on an interval $[x_i, x_{i+1}]$ depends only on the points (x_i, y_i) and (x_{i+1}, y_{i+1}) and the slopes m_i and m_{i+1} at these points. The subroutine determines which of Cases 1–4 applies [3]. The tolerance parameter EPS is used by CHOOSE to handle the pathologies that may occur. When the user provides values for the slopes other than those computed by the subroutine SLOPES, small perturbations may cause unwanted changes in the shape of the resulting spline. If the values of m_i or m_{i+1} are relatively close to either the slope S_i of the line joining the points (x_i, y_i) and (x_{i+1}, y_{i+1}) or to $2S_i$, roundoff may affect both the monotonicity and convexity of the spline. For example, if $|M - S_i| \leq \text{EPS}\,|S_i|$ for $M = m_i$ or m_{i+1}, then the algorithm assumes that $M = S_i$ and chooses either Case 2 or Case 3. We avoid Case 1 to preclude a nearly linear spline joining the two data points. Similarly, in order to avoid a spline with sharp bends at the endpoints, if $|m_i| > 2|S_i|$ and $|m_{i+1}| > (2 - \text{EPS})|S_i|$, then the algorithm assumes $|m_{i+1}| > 2|S_i|$ and selects Case 4 rather than Case 3. These choices are for aesthetics only and can easily be overridden by modifying the slopes m_i and m_{i+1} and/or setting EPS to a smaller value. If EPS $\neq 0$, then EPS should be greater than or equal to machine epsilon.

Subroutine CASES

After CHOOSE determines the correct case number, the subroutine CASES calculates the knots and other parameters which define S on the interval $[x_i, x_{i+1}]$. The spline S will have a single knot between x_i and x_{i+1} if NCASE is 1, 2, or 3. Case 4 requires two knots.

Function Subprogram SPLINE

Given the case number determined by CHOOSE and the parameters calculated by CASES, SPLINE determines the location of the argument relative to the knots in the interval and computes $S(u_j)$, which is the value of a quadratic Bernstein polynomial.

ACKNOWLEDGMENTS

The authors would like to thank Steven L. Dodd and Marie Roulier of North Carolina State University for their help in coding the algorithm.

REFERENCES

(Note. References [1, 2, 4, 5, 6, 7] are not cited in the text.)

1. DEIMEL, L.E., DOSS, C.L., FORNARO, R.J., MCALLISTER, D.F., AND ROULIER, J.A. Application of shape-preserving spline interpolation to interactive editing of photogrammetric data. *Proc. SIGGRAPH* '78, vol. 12, no. 3, pp. 93–99.
2. DEIMEL, L.E., MCALLISTER, D.F., AND ROULIER, J.A. Smooth curve fitting with shape preservation using osculatory quadratic splines. *Proc. 11th Annu. Conf. on the Interface Between Statistics and Computer Science*, 1978, pp. 343–347.
3. MCALLISTER, D.F., AND ROULIER, J.A. An algorithm for computing a shape-preserving osculatory quadratic spline. *ACM Trans. Math. Softw. 7,* 3(Sept. 1981), 331–347.
4. MCALLISTER, D.F., AND ROULIER, J.A. Interpolation by convex quadratic splines. *Math. Comp. 32,* 144 (1978), pp. 1154–1162.
5. MCALLISTER, D.F., AND ROULIER, J.A. Approximation by convex quadratic splines. *Approximation Theory III.* Academic Press, 1980, pp. 757–761.
6. MCALLISTER, D.F., PASSOW, E., AND ROULIER, J.A. Algorithms for computing shape preserving spline interpolations to data. *Math. Comp. 31,* 139 (July 1977), 717–725.
7. MCALLISTER, D.F., ROULIER, J.A., AND EVANS, M. Generation of random numbers using shape preserving quadratic splines. *Proc. 16th Annu. Southeast Regional ACM Conference*, April 1978, pp. 216–218.

ALGORITHM

[The complete listing is available from the ACM Algorithms Distribution Service .]

ALGORITHM 575
Permutations for a Zero-Free Diagonal [F1]

I.S. Duff
AERE Harwell, England

Key Words and Phrases: unsymmetric permutations, maximum transversal, maximum assignment, block triangular form, sparse matrices
CR Categories: 5.0, 5.1, 5.3, 5.4
Language: FORTRAN

DESCRIPTION

The subroutine is evoked by the FORTRAN statement

CALL MC21A(N, ICN, LICN, IP, LENR, IPERM, NUMNZ, IW)

where the parameters are described in the listing given here.

Given the pattern of nonzeros of a sparse matrix, this subroutine attempts to find a row permutation that makes the matrix have no zeros on its diagonal. It is possible that the user may input a matrix for which there is no permutation that makes the diagonal zero-free. An example of this is

$$\begin{pmatrix} X & 0 \\ X & 0 \end{pmatrix}$$

In such instances the algorithm will produce a permutation that will put as many nonzeros on the diagonal as possible (1 in the above example). This number will be output in NUMNZ. The array IPERM will still hold a permutation of the integers 1, 2, . . . , N, but in this case N-NUMNZ of the elements (IPERM(I), I) will be zero.

It is envisaged that a common use of the subroutine will be as the first part of a two-stage process for the determination of the block triangular form of a sparse matrix (see, for example, [1]). The second stage could be performed by Harwell subroutine MC13D [4], and the two stages are combined by using subroutine MC23A in the Harwell package MA28 [2].

The subroutine is based on a depth first search with look-ahead technique and is described in detail in [3]. There, numerical results from using this subroutine, which has been tested on a wide range of both structured and randomly generated matrices, are also discussed.

This subroutine has been written in American National Standard FORTRAN. Special comment cards have been included so that Harwell subroutine OE04A can be used to make an IBM FORTRAN version that uses half-length integers (INTEGER*2) for all the arrays except IP. This approximately halves the core requirements at the cost of restricting the order of the system to $2^{15} - 1$.

Received 24 October 1978, revised 10 January 1980, accepted April 1980.

Author's address: Computer Science and Systems Division, Building 8.9, AERE Harwell, Oxon., England.

REFERENCES

1. DUFF, I.S. On permutations to block triangular form. *J. Inst. Math. Appl. 19* (1977), 339–342.
2. DUFF, I.S. MA28—A set of Fortran subroutines for sparse unsymmetric linear equations. AERE Rep. R.8730, Her Majesty's Stationery Office, London, 1977.
3. DUFF, I.S. On algorithms for obtaining a maximum transversal. *ACM Trans. Math. Softw. 7,* 3 (Sept. 1981), 315–330.
4. DUFF, I.S., AND REID, J.K. Algorithm 529. Permutations to block triangular form. *ACM Trans. Math. Softw. 4,* 2 (June 1978), 189–192.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service.]

```
C S IS THE STANDARD FORTRAN VERSION   SI/
C I IS THE IBM VERSION
C
      SUBROUTINE MC21A(N, ICN, LICN, IP, LENR, IPERM, NUMNZ, IW)
C
C DESCRIPTION OF PARAMETERS.
C INPUT VARIABLES   N, ICN, LICN, IP, LENR.
C OUTPUT VARIABLES  IPERM, NUMNZ.
C
C N   ORDER OF MATRIX.
C ICN ARRAY CONTAINING THE COLUMN INDICES OF THE NON-ZEROS.  THOSE
C     BELONGING TO A SINGLE ROW MUST BE CONTIGUOUS BUT THE ORDERING
C     OF COLUMN INDICES WITHIN EACH ROW IS UNIMPORTANT AND WASTED
C     SPACE BETWEEN ROWS IS PERMITTED.
C LICN  LENGTH OF ARRAY ICN.
C IP  IP(I), I=1,2,...N, IS THE POSITION IN ARRAY ICN OF THE FIRST
C     COLUMN INDEX OF A NON-ZERO IN ROW I.
C LENR  LENR(I) IS THE NUMBER OF NON-ZEROS IN ROW I, I=1,2,..N.
C IPERM CONTAINS PERMUTATION TO MAKE DIAGONAL HAVE THE SMALLEST
C     NUMBER OF ZEROS ON IT.  ELEMENTS (IPERM(I),I) I=1, ... N ARE
C     NON-ZERO AT THE END OF THE ALGORITHM UNLESS MATRIX
C     IS STRUCTURALLY SINGULAR.  IN THIS CASE, (IPERM(I),I) WILL
C     BE ZERO FOR N-NUMNZ ENTRIES.
C NUMNZ NUMBER OF NON-ZEROS ON DIAGONAL OF PERMUTED MATRIX.
C IW  WORK ARRAY  ..  SEE LATER COMMENTS.
C
      INTEGER IP(N)
C     INTEGER*2 ICN(LICN),LENR(N),IPERM(N),IW(N,4)     I/
      INTEGER ICN(LICN), LENR(N), IPERM(N), IW(N,4)
      CALL MC21B(N, ICN, LICN, IP, LENR, IPERM, NUMNZ, IW(1,1),
     * IW(1,2), IW(1,3), IW(1,4))
      RETURN
      END
C/
      SUBROUTINE MC21B(N, ICN, LICN, IP, LENR, IPERM, NUMNZ, PR, ARP,
     * CV, OUT)
      INTEGER IP(N)
C
C     DIVISION OF WORK ARRAY IS NOW DESCRIBED.
C
C PR(I) IS THE PREVIOUS ROW TO I IN THE DEPTH FIRST SEARCH.
C ARP(I) IS ONE LESS THAN THE NUMBER OF NON-ZEROS IN ROW I
C     WHICH HAVE NOT BEEN SCANNED WHEN LOOKING FOR A CHEAP ASSIGNMENT.
C CV(I) IS THE MOST RECENT ROW EXTENSION AT WHICH COLUMN I
C     WAS VISITED.
C OUT(I) IS ONE LESS THAN THE NUMBER OF NON-ZEROS IN ROW I
C     WHICH HAVE NOT BEEN SCANNED DURING ONE PASS THROUGH THE
C     MAIN LOOP.
C
C     INTEGER*2 ICN(LICN),LENR(N),IPERM(N),PR(N),CV(N),   I/
C    1ARP(N),OUT(N)  I/
      INTEGER ICN(LICN), LENR(N), IPERM(N), PR(N), CV(N), ARP(N), OUT(N)
C
C   INITIALIZATION OF ARRAYS.
      DO 10 I=1,N
        ARP(I) = LENR(I) - 1
        CV(I) = 0
        IPERM(I) = 0
   10 CONTINUE
```

```
      NUMNZ = 0
C
C
C   MAIN LOOP.
C   EACH PASS ROUND THIS LOOP EITHER RESULTS IN A NEW ASSIGNMENT
C OR GIVES A ROW WITH NO ASSIGNMENT.
      DO 100 JORD=1,N
        J = JORD
        PR(J) = -1
        DO 70 K=1,JORD
C LOOK FOR A CHEAP ASSIGNMENT
          IN1 = ARP(J)
          IF (IN1.LT.0) GO TO 30
          IN2 = IP(J) + LENR(J) - 1
          IN1 = IN2 - IN1
          DO 20 II=IN1,IN2
            I = ICN(II)
            IF (IPERM(I).EQ.0) GO TO 80
   20     CONTINUE
C   NO CHEAP ASSIGNMENT IN ROW.
          ARP(J) = -1
C   BEGIN LOOKING FOR ASSIGNMENT CHAIN STARTING WITH ROW J.
   30     OUT(J) = LENR(J) - 1
C INNER LOOP.  EXTENDS CHAIN BY ONE OR BACKTRACKS.
          DO 60 KK=1,JORD
            IN1 = OUT(J)
            IF (IN1.LT.0) GO TO 50
            IN2 = IP(J) + LENR(J) - 1
            IN1 = IN2 - IN1
C FORWARD SCAN.
            DO 40 II=IN1,IN2
              I = ICN(II)
              IF (CV(I).EQ.JORD) GO TO 40
C   COLUMN I HAS NOT YET BEEN ACCESSED DURING THIS PASS.
              J1 = J
              J = IPERM(I)
              CV(I) = JORD
              PR(J) = J1
              OUT(J1) = IN2 - II - 1
              GO TO 70
   40       CONTINUE
C
C   BACKTRACKING STEP.
   50       J = PR(J)
            IF (J.EQ.-1) GO TO 100
   60     CONTINUE
C
   70   CONTINUE
C
C   NEW ASSIGNMENT IS MADE.
   80   IPERM(I) = J
        ARP(J) = IN2 - II - 1
        NUMNZ = NUMNZ + 1
        DO 90 K=1,JORD
          J = PR(J)
          IF (J.EQ.-1) GO TO 100
          II = IP(J) + LENR(J) - OUT(J) - 2
          I = ICN(II)
          IPERM(I) = J
   90   CONTINUE
C
  100 CONTINUE
C
C   IF MATRIX IS STRUCTURALLY SINGULAR, WE NOW COMPLETE THE
C PERMUTATION IPERM
      IF (NUMNZ.EQ.N) GO TO 150
      DO 110 I=1,N
        ARP(I) = 0
  110 CONTINUE
      K = 0
      DO 130 I=1,N
        IF (IPERM(I).NE.0) GO TO 120
        K = K + 1
        OUT(K) = I
        GO TO 130
  120   J = IPERM(I)
```

```
      ARP(J) = 1
130 CONTINUE
    K = 0
    DO 140 I=1,N
      IF (ARP(I).NE.0) GO TO 140
      K = K + 1
      IOUTK = OUT(K)
      IPERM(IOUTK) = I
140 CONTINUE
150 RETURN
    END
```

ALGORITHM 576
A FORTRAN Program for Solving $\mathbf{Ax} = \mathbf{b}$ [F4]

I. BARRODALE
University of Victoria
and
G. F. STUART
Camosun College

Key Words and Phrases: linear equations, Gaussian elimination, new pivoting strategy
CR Categories: 5.13, 5.14
Language: FORTRAN

DESCRIPTION

Given any $n \times n$ system of linear equations

$$\mathbf{Ax} = \mathbf{b} \tag{1}$$

and a number ϵ, the algorithm calculates the solution $\mathbf{x}$ to (1), if $\epsilon \leq 0$, or an approximation solution $\mathbf{x}^{(k)}$ satisfying

$$\| \mathbf{b} - \mathbf{Ax}^{(k)} \|_\infty < \epsilon, \tag{2}$$

if $\epsilon > 0$. Furthermore, if $\mathbf{A}$ appears to be singular, an approximate solution to (1) is still determined, although it does not satisfy (2).

The algorithm consists of Gaussian elimination combined with a new pivoting strategy based on the following fact. Suppose after the kth step ($k = 0, 1, \ldots, n - 1$) of Gaussian elimination that $\mathbf{x}^{(k)} = [x_1^{(k)}, \ldots, x_k^{(k)}, 0, \ldots, 0]^T$ is the approximate solution obtained by back substitution in the first k equations. *Then the right-hand sides of the last $n - k$ equations are the nonzero components of the residual vector* $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{Ax}^{(k)}$; this is proved in [1]. Our pivoting strategy thus chooses the equation with the absolutely largest residual as the next equation to be eliminated, and the $(k + 1)$th pivot is then selected as the absolutely largest of the $n - k$ coefficients on the left-hand side of the chosen equation. Finally, the pivot is positioned on the principal diagonal by, if necessary, a row interchange and a column interchange.

When $\epsilon > 0$, the algorithm terminates after k Gaussian elimination steps and yields $\mathbf{x}^{(k)}$ by back substitution in the first k equations. Normally, $\mathbf{x}^{(k)}$ has k nonzero components and it satisfies k of the eqs. (1) exactly (ignoring rounding errors). However, if ϵ is a *small* positive number, in order to satisfy (2) it may be necessary to determine the solution $\mathbf{x}$ rather than $\mathbf{x}^{(k)}$: in such a case, k has the value n on output.

Received 13 July 1976, revised 22 April 1980, accepted 13 May 1980.

This work was supported by National Research Council of Canada Grant A-5251.
Authors' addresses: I. Barrodale, Department of Mathematics, University of Victoria, Victoria, B.C., Canada; G.F. Stuart, Camosun College, Victoria, B.C., Canada.

The elimination is performed by subroutine MODGE, which terminates after stage k if there is not a suitably large residual among the remaining $(n - k)$ right-hand sides, or if the coefficient matrix of the remaining $(n - k)$ rows is considered to be singular. The first condition is detected by comparing all residuals with the real input parameter EPS. Hence EPS should be set to a small positive value representing an insignificant residual for cases in which premature termination is sought; otherwise EPS should be set less than or equal to zero. The second premature termination condition is controlled by the real input parameter TOLER. The equation with the absolutely largest residual is rejected as the pivot row if the absolute value of the pivot is not greater than TOLER. In this case complete pivoting is performed during the $(k + 1)$th stage. However, if this produces no pivot greater in absolute value than TOLER, the elimination is terminated and an approximate solution is determined by back substitution in the first k equations.

The algorithm can thus be applied to any $n \times n$ system of linear equations, although it is particularly well suited to problems where the residuals (right-hand sides) can be made small by solving for fewer than n of the unknowns. Interpolation problems are often of this type, since some of the parameters x_j frequently turn out to be quite small in practice. Also, ill-conditioned (and even singular) systems of equations can often be handled satisfactorily by choosing appropriate values for EPS and TOLER (see Examples 3 and 4).

The accuracy of a computed solution to $\mathbf{Ax} = \mathbf{b}$ can usually be increased by iterative improvement, and subroutine REFINE is supplied for this purpose. A feature of this subroutine is that the $(t + 1)$th residual vector $\mathbf{r}_{t+1}$ is computed as

$$\mathbf{r}_{t+1} = \mathbf{r}_t - \mathbf{A}\mathbf{x}_t, \qquad t = 0, 1, 2, \ldots \tag{3}$$

where the residual $\mathbf{r}_t$ and the computed solution $\mathbf{x}_t$ are taken from the previous iteration, and $\mathbf{r}_0 = \mathbf{b}$. (It has been our experience that (3) permits better control of convergence than the more usual computation $\mathbf{r}_{t+1} = \mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{x}_1 + \cdots + \mathbf{x}_t)$.) In this subroutine the output parameter DIGITS is used to estimate (usually fairly conservatively) the number of significant digits in the initial solution $\mathbf{x}_0$. Basically, this is determined as $\log_{10} D$, where $D = \min_i |x_{0,i}/x_{1,i}|$, excluding values of i for which $x_{0,i} = 0$ or $x_{1,i} = 0$. However, if the quantity $\log_{10} D$ is greater than $\log_{10}(1/E)$, where $E = \max(\text{ERR}, 10^{-d})$ and d represents the number of decimal digits of accuracy available on the computer, then DIGITS is set equal to $\log_{10}(1/E)$ rather than $\log_{10} D$.

The following four examples were solved on an IBM 370/148 using a WATFIV compiler. All of the computations were performed in single-precision arithmetic (about 7 decimal digits) apart from the calculation of the residuals (3), which were formed in double-precision arithmetic and then assigned to a single-precision vector.

Example 1. Determine parameters x_1, x_2, and x_3 such that the quadratic polynomial $x_1 + x_2 t + x_3 t^2$ interpolates the data {(1, 89), (4, 209), (5, 201)}. Equivalently, solve the 3×3 system of linear equations

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 89 \\ 209 \\ 201 \end{bmatrix}.$$

This example is used in [1] to illustrate our algorithm: the exact solution is $x_1 = 1$, $x_2 = 100$, and $x_3 = -12$.

Setting EPS = 0 and TOLER = 10^{-6}, subroutine MODGE gives $x_1 = 1.000010$, $x_2 = 99.99998$, and $x_3 = -11.99999$.

Setting EPS = 1 and TOLER = 10^{-6}, subroutine MODGE terminates after two Gaussian elimination steps and gives $x_1 = 0$, $x_2 = 100.4500$, and $x_3 = -12.04999$.

Example 2. Determine $x_1, x_2, \ldots, x_5$ such that the function $x_1 t + x_2 t^2 + x_3 t^3 + x_4 t^4 + x_5 t^5$ interpolates te^{-t} for $t = 0.25(0.25)1.25$. Equivalently, solve $\mathbf{Ax} = \mathbf{b}$ where

$$\mathbf{A} = \begin{bmatrix} 0.25 & 0.25^2 & \cdots & 0.25^5 \\ 0.50 & 0.50^2 & \cdots & 0.50^5 \\ 0.75 & 0.75^2 & \cdots & 0.75^5 \\ 1.00 & 1.00^2 & \cdots & 1.00^5 \\ 1.25 & 1.25^2 & \cdots & 1.25^5 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 0.25e^{-0.25} \\ 0.50e^{-0.50} \\ 0.75e^{-0.75} \\ 1.00e^{-1.00} \\ 1.25e^{-1.25} \end{bmatrix}.$$

Generating the elements of **A** and **b** within the computer (rather than reading them in as data), and setting EPS = 0 and TOLER = 10^{-6}, subroutine MODGE gives $x_1 = 0.9994718$, $x_2 = -0.9950786$, $x_3 = 0.4832436$, $x_4 = -0.1396533$, and $x_5 = 0.01989542$. (This solution contains almost one extra significant digit than the solution that can be obtained from Gaussian elimination with partial pivoting, using the modified version [3] of the subroutines in [2].) Iterative improvement is then applied to this initial solution.

Setting ERR = 10^{-7} and ITER = 14, subroutine REFINE converges after two iterations to $x_1 = 0.9994708$, $x_2 = -0.9950703$, $x_3 = 0.4832247$, $x_4 = -0.1396357$, and $x_5 = 0.01988983$. The output parameter DIGITS indicates that the initial solution contains approximately 3.6 significant digits.

Example 3. Determine $x_1, x_2, \ldots, x_9$ such that the function $x_1 + x_2t + x_3(t-1) + x_4t^2 + x_5(t^2 - t) + x_6t^3 + x_7(t^3 - t^2) + x_8t^4 + x_9(t^4 - t^3)$ interpolates e^t for $t = 0(0.125)1$.

Setting EPS = TOLER = 10^{-6}, subroutine MODGE gives $x_1 = 0.9999993$, $x_2 = 0.9994056$, $x_3 = 0$, $x_4 = 0.5071687$, $x_5 = x_6 = x_7 = 0$, $x_8 = 0.2117074$, and $x_9 = -0.1439046$. The output parameters OCODE and PIVOT have the values 3 and 2, respectively, indicating that the computation was terminated prematurely because a pivot was chosen (by complete pivoting) whose magnitude is less than TOLER.

Setting ERR = 10^{-7} and ITER = 14, subroutine REFINE converges after two iterations to $x_1 = 1.000000$, $x_2 = 0.9994109$, $x_4 = 0.5071452$, $x_8 = 0.2117255$, and $x_9 = -0.1439375$. DIGITS has the value 3.6 on output.

Example 4. Determine $x_1, x_2, \ldots, x_{32}$ such that the function

$$P(s, t) = x_1 + \sum_{j=1}^{15} (x_{2j} \operatorname{Re}(z^j) + x_{2j+1} \operatorname{Im}(z^j)) + x_{32} \operatorname{Re}(z^{16}),$$

where $z = s + it$, interpolates $Q(s, t) = e^{s^2 - st + 2t^2}$ on the 32 points (s, t) for which $s = 0(0.125)1$, $t = 0(0.125)1$. This problem arises when attempting to solve approximately the Dirichlet problem for Laplace's equation on a unit square with boundary value Q. The coefficients x_j of the harmonic polynomial P are determined by collocation, that is, by requiring that $P = Q$ on the 32 specified boundary points. (We arranged the resulting 32 linear equations in order by proceeding in an anticlockwise direction around the boundary of the square, starting from the origin.)

Setting EPS = 0 and TOLER = 10^{-6}, subroutine MODGE gives an initial solution containing *no* significant digits: many of the 32 computed coefficients x_j have exponents of size 10^3 or 10^4 and only eight coefficients have magnitudes less than 20. (The initial solution that can be obtained from Gaussian elimination with partial pivoting also contains no significant digits.) Setting ERR = 10^{-7} and ITER = 14, subroutine REFINE diverges, that is, all 14 iterations are performed without improvement in the accuracy of the coefficients x_j. The output parameter DIGITS has a *negative* value, clearly indicating that the initial solution contains no significant digits.

However, setting EPS = 0.05 and TOLER = 10^{-6}, subroutine MODGE terminates after 22 Gaussian elimination steps and gives an initial approximate solution in which 10 coefficients are set to zero and the remaining 22 coefficients vary from a minimum of $x_{27} = -9.698888$ to a maximum of $x_{29} = 12.78670$. Setting ERR = 10^{-7} and ITER = 14, subroutine REFINE converges after three iterations: in this final solution $x_{27} = -9.699727$ and $x_{29} = 12.78701$. The output parameter

DIGITS indicates that the initial approximate solution contains approximately 2.9 significant digits.

This example illustrates how our algorithm can be used to produce a meaningful approximate solution to a system of linear equations whose numerical solution may be difficult to determine. The harmonic polynomial P^*, say, given by the above approximate solution satisfies $\| Q - P^* \|_\infty \leq 0.05$ on the 32 specified boundary bounds, and a fine grid search establishes that $\| Q - P^* \|_\infty \doteq 0.06$ on the complete boundary. By the maximum principle, it follows that the latter inequality also applies on the whole square.

ACKNOWLEDGMENTS

We wish to thank both the referees for their constructive criticisms of this paper and Mr. K.B. Wilson for his programming assistance.

REFERENCES

(Note. References [4, 5] are not cited in the text.)

1. BARRODALE, I., AND STUART, G. A new variant of Gaussian elimination. *J. Inst. Math. Appl. 19* (1977), 39–47.
2. FORSYTHE, G., AND MOLER, C.B. *Computer Solution of Linear Algebraic Systems.* Prentice-Hall, Englewood Cliffs, N.J., 1967.
3. LAWSON, C.L., HANSON, R.J., KINCAID, D.R., AND KROGH, F.T. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Softw. 5*, 3 (Sept. 1979), 308–323.
4. LAWSON, C.L., HANSON, R.J., KINCAID, D.R., AND KROGH, F.T. Algorithm 539. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Softw. 5*, 3 (Sept. 1979), 324–325.
5. MOLER, C.B. Matrix computations with Fortran and paging. *Commun. ACM 15*, 4 (April 1972), 268–270.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service.]

```
      SUBROUTINE MODGE(N, NDIM, A, B, EPS, TOLER, P, Q, Z,
     * X, K, OCODE, OPIVOT)
C THIS SUBROUTINE SOLVES ANY N-BY-N SYSTEM OF LINEAR
C EQUATIONS AX=B, USING A MODIFICATION OF GAUSSIAN
C ELIMINATION. THE MODIFICATION OCCURS IN THE CHOICE OF
C PIVOTS: AT EACH STAGE THE PIVOT ROW IS DETERMINED BY THE
C ABSOLUTELY LARGEST RESIDUAL (RIGHT HAND SIDE), AND THEN
C THE PIVOT IS CHOSEN AS THE ABSOLUTELY LARGEST COEFFICIENT
C IN THE PIVOT ROW. THE PIVOT IS POSITIONED ON THE PRINCIPAL
C DIAGONAL BY (IF NECESSARY) A ROW INTERCHANGE AND A COLUMN
C INTERCHANGE.
C
C IF AFTER THE K-TH STAGE THE LARGEST RESIDUAL IS TOO SMALL
C (.LT.EPS), THE ELIMINATION IS TERMINATED. BACK
C SUBSTITUTION IS THEN USED TO OBTAIN AN APPROXIMATE
C SOLUTION WHICH SATISFIES THE FIRST K EQUATIONS.
C
C IF THE (K+1)-TH PIVOT IS TOO SMALL (.LE.TOLER), THE
C ELIMINATION IS TERMINATED. BACK SUBSTITUTION IS THEN USED
C TO OBTAIN AN APPROXIMATE SOLUTION WHICH SATISFIES THE
C FIRST K EQUATIONS.
C
C DESCRIPTION OF PARAMETERS:
C N       AN INTEGER DENOTING THE NUMBER OF EQUATIONS
C         (N.GE.1).
C NDIM    AN INTEGER PARAMETER FOR ADJUSTABLE DIMENSIONS
C         (NDIM.GE.N).
C A       A REAL ARRAY OF NDIM ROWS AND AT LEAST N
C         COLUMNS WHICH ON ENTRY MUST CONTAIN THE COEFFICIENT
C         MATRIX A IN ITS FIRST N ROWS AND N COLUMNS.
C         ON EXIT THE FIRST K ROWS AND K COLUMNS (K.LE.N) OF
C         A CONTAIN THE LU DECOMPOSITION OF THE K EQUATIONS
C         SATISFIED.
C B       A REAL ARRAY OF DIMENSION AT LEAST N WHICH
```

```
C          ON ENTRY MUST CONTAIN IN POSITIONS 1 TO N THE RIGHT
C          HAND SIDE VECTOR B.
C          ON EXIT THE POSITIONS K+1 TO N CONTAIN THE RESIDUALS
C          OF THE UNSATISFIED EQUATIONS.
C EPS      A REAL INPUT TOLERANCE. IF THE (ABSOLUTELY) LARGEST
C          RESIDUAL AT ANY STAGE OF THE ELIMINATION IS LESS
C          THAN EPS THEN THE ELIMINATION IS TERMINATED AND
C          BACK SUBSTITUTION COMMENCES. PREMATURE TERMINATION
C          OF THIS TYPE CAN BE AVOIDED BY SETTING EPS TO ZERO.
C TOLER    A REAL POSITIVE INPUT TOLERANCE. IF THE ABSOLUTE
C          VALUE OF THE PIVOT AT ANY STAGE OF THE ELIMINATION
C          IS LESS THAN OR EQUAL TO TOLER, THEN COMPLETE
C          PIVOTING IS USED FOR THIS STAGE. IF THE ABSOLUTE
C          VALUE OF THE NEW PIVOT IS LESS THAN OR EQUAL TO
C          TOLER, THE REMAINING EQUATIONS ARE CONSIDERED
C          TO BE LINEARLY DEPENDENT, AND SO THE ELIMINATION
C          IS TERMINATED AND BACK SUBSTITUTION COMMENCES.
C          TOLER SHOULD NORMALLY BE SET TO APPROXIMATELY
C          10**(-D+1), WHERE D REPRESENTS THE NUMBER OF
C          DECIMAL DIGITS OF ACCURACY AVAILABLE. HOWEVER,
C          PROBLEMS WHICH ARE NOT 'REASONABLY SCALED'
C          BEFOREHAND COULD REQUIRE A MUCH LARGER OR
C          SMALLER VALUE FOR TOLER THAN IS RECOMMENDED
C          HERE.
C P        AN INTEGER ARRAY OF DIMENSION AT LEAST N USED TO
C          RECORD COLUMN INTERCHANGES OF THE ARRAY A.
C          ON EXIT P(I) IS THE INDEX OF THE ORIGINAL POSITION
C          OF COLUMN I.
C Q        AN INTEGER ARRAY OF DIMENSION AT LEAST N USED TO
C          RECORD ROW INTERCHANGES OF THE ARRAYS A AND B.
C          ON EXIT Q(I) IS THE INDEX OF THE ORIGINAL POSITION
C          OF ROW I.
C Z        A REAL WORK SPACE ARRAY OF DIMENSION AT LEAST N.
C X        A REAL OUTPUT ARRAY OF DIMENSION AT LEAST N
C          WHICH CONTAINS THE SOLUTION (IN CORRECT ORDER) OF
C          THE N LINEAR EQUATIONS IN POSITIONS 1 TO N.
C          IF THE SOLUTION IS OBTAINED USING LESS THAN N
C          EQUATIONS, THEN THE REMAINING COMPONENTS OF X ARE
C          SET TO ZERO.
C K        AN INTEGER WHICH ON OUTPUT DENOTES THE NUMBER OF
C          EQUATIONS SATISFIED (K.LE.N).
C OCODE    AN INTEGER EXIT CODE WITH VALUES:
C            0- EITHER N.LE.0 OR (N.EQ.1 AND A(1,1).EQ.0).
C            1- THE SYSTEM AX=B IS SOLVED.
C            2- THE ELIMINATION IS TERMINATED BY USE OF
C               EPS (SMALL RESIDUAL).
C            3- THE ELIMINATION IS TERMINATED BY USE OF
C               TOLER (SMALL PIVOT).
C OPIVOT AN INTEGER EXIT CODE WITH VALUES:
C            1- COMPLETE PIVOTING IS NOT EMPLOYED.
C            2- COMPLETE PIVOTING IS EMPLOYED, HENCE SMALL
C               RESIDUALS (.LT.EPS) MAY HAVE BEEN ALLOWED.
C
C AN IMPLEMENTATION NOTE:
C IF YOUR FORTRAN COMPILER PERMITS A SINGLE COLUMN OF A TWO
C DIMENSIONAL ARRAY TO BE PASSED TO A ONE DIMENSIONAL ARRAY
C THROUGH A SUBROUTINE CALL, CONSIDERABLE SAVINGS IN
C EXECUTION TIME MAY BE ACHIEVED BY USING SUBROUTINE
C SAXPY (SEE A.C.M. TRANSACTIONS ON MATHEMATICAL
C SOFTWARE VOL.5 NUM.3 1979 PP.308-323)
      SUBROUTINE REFINE(N, NDIM, A, B, ALU, P, Q, KMAX,
     * ERR, R, S, XNEW, DX, X, ITER, DIGITS)
C THIS SUBROUTINE PERFORMS ITERATIVE IMPROVEMENT ON THE
C COMPUTED SOLUTION X OBTAINED FROM SUBROUTINE MODGE.
C THE ITERATIONS CONTINUE UNTIL EITHER THE CORRECTIONS
C TO X ARE NEGLIGIBLE, I.E. CONVERGENCE OCCURS (SEE
C DESCRIPTION OF ERR), OR UNTIL COMPLETION OF THE
C MAXIMUM NUMBER OF ITERATIONS ALLOWED (ITER).
C
C THE FOLLOWING TABLE GIVES THE CORRESPONDENCE BETWEEN THE
C OUTPUT PARAMETERS OF MODGE AND THE INPUT PARAMETERS OF
```

```
C REFINE:
C         NAME ON OUTPUT FROM MODGE:  NAME ON INPUT TO REFINE:
C         N                           N
C         NDIM                        NDIM
C         A                           ALU
C         P                           P
C         Q                           Q
C         K                           KMAX
C         X                           X
C
C DESCRIPTION OF PARAMETERS:
C N       AN INTEGER DENOTING THE NUMBER OF EQUATIONS
C         (N.GE.1).
C NDIM    AN INTEGER PARAMETER FOR ADJUSTABLE DIMENSIONS
C         (NDIM.GE.N).
C A       A REAL ARRAY OF NDIM ROWS AND AT LEAST N COLUMNS
C         WHICH ON ENTRY MUST CONTAIN THE COEFFICIENT
C         MATRIX A IN ITS FIRST N ROWS AND N COLUMNS.
C         A IS NOT ALTERED BY REFINE.
C B       A REAL ARRAY OF DIMENSION AT LEAST N WHICH ON
C         ENTRY MUST CONTAIN IN POSITIONS 1 TO N THE RIGHT
C         HAND SIDE VECTOR B.  B IS NOT ALTERED BY REFINE.
C ALU     THE REAL OUTPUT ARRAY A OF SUBROUTINE MODGE.  ALU IS
C         NOT ALTERED BY REFINE.
C P       AN INTEGER ARRAY OF DIMENSION AT LEAST N, USED
C         TO RECORD COLUMN INTERCHANGES OF THE ARRAY A.
C         P IS THE OUTPUT ARRAY P OF SUBROUTINE MODGE.
C Q       AN INTEGER ARRAY OF DIMENSION AT LEAST N, USED
C         TO RECORD ROW INTERCHANGES OF THE ARRAYS A AND B.
C         Q IS THE OUTPUT ARRAY Q OF SUBROUTINE MODGE.
C KMAX    THE INTEGER OUTPUT PARAMETER K OF MODGE.
C ERR     A REAL INPUT PARAMETER USED TO TEST FOR
C         CONVERGENCE.  ERR SHOULD NOT BE LESS THAN 10**(-D),
C         WHERE D REPRESENTS THE NUMBER OF DECIMAL DIGITS
C         OF ACCURACY AVAILABLE.
C         THE CONVERGENCE CRITERION IS:
C         EXIT IF THE FOLLOWING RELATIONSHIP IS TRUE FOR
C         ALL I=1 TO KMAX:
C         ABS(DX(I)).LE.ABS(ERR*X(I)).OR.X(I).EQ.0.0 .
C R       A REAL WORK SPACE ARRAY OF DIMENSION AT LEAST N.
C         (USED BY REFINE TO CALCULATE THE RESIDUALS AT EACH
C         ITERATION.)
C S       A REAL WORK SPACE ARRAY OF DIMENSION AT LEAST N.
C         (USED BY REFINE TO CALCULATE THE CORRECTIONS, DX,
C         AT EACH ITERATION.)
C XNEW    A REAL WORK SPACE ARRAY OF THE SAME DIMENSION AS X.
C DX      A REAL WORK SPACE ARRAY OF DIMENSION AT LEAST N.
C         (USED BY REFINE TO CALCULATE A CORRECTION TO X
C         AT EACH ITERATION.)
C X       THE REAL OUTPUT ARRAY X OF SUBROUTINE MODGE.
C         ON EXIT X CONTAINS (USUALLY) A BETTER SOLUTION
C         TO THE LINEAR SYSTEM AX=B.
C ITER    AN INTEGER DENOTING, ON INPUT, THE MAXIMUM NUMBER
C         OF ITERATIONS OF ITERATIVE IMPROVEMENT ALLOWED.
C         ITER SHOULD NOT EXCEED 2*D, WHERE D REPRESENTS
C         THE NUMBER OF DECIMAL DIGITS OF ACCURACY AVAILABLE.
C         ON OUTPUT ITER IS THE ACTUAL NUMBER OF ITERATIONS
C         COMPLETED.
C DIGITS A REAL PARAMETER WHICH ON OUTPUT INDICATES THE
C         APPROXIMATE NUMBER OF SIGNIFICANT DIGITS IN THE
C         INITIAL SOLUTION (I.E.  THE SOLUTION FROM
C         MODGE).
```

ALGORITHM 577
Algorithms for Incomplete Elliptic Integrals [S21]

B. C. CARLSON and ELAINE M. NOTIS
Iowa State University

Key Words and Phrases: elliptic integrals, logarithms, inverse circular functions, inverse hyperbolic functions, *R*-functions
CR Categories: 5.12
Language: FORTRAN

DESCRIPTION

Robust and highly portable FORTRAN programs, implementing algorithms proved in [1], are presented for the symmetric elliptic integral of the first kind,

$$R_F(x, y, z) = \frac{1}{2}\int_0^\infty [(t + x)(t + y)(t + z)]^{-1/2}\, dt, \tag{1}$$

and a symmetric elliptic integral of the third kind,

$$R_J(x, y, z, \rho) = \frac{3}{2}\int_0^\infty [(t + x)(t + y)(t + z)]^{-1/2}(t + \rho)^{-1}\, dt. \tag{2}$$

All arguments are nonnegative, and if $x = 0$, the integrals are complete. The accessible range of argument values can be extended beyond the range admissible in the programs by using homogeneity:

$$R_F(kx, ky, kz) = k^{-1/2}R_F(x, y, z), \qquad R_J(kx, ky, kz, k\rho) = k^{-3/2}R_J(x, y, z, \rho).$$

The range of function values is indicated by

$$M^{-1/2} \le R_F(x, y, z) < 2m^{-1/2}, \qquad M_1^{-3/2} \le R_J(x, y, z, \rho) < 5m_1^{-3/2},$$
$$M = \max\{x, y, z\}, \qquad M_1 = \max\{M, \rho\},$$
$$m = \min\{x + y, x + z, y + z\}, \qquad m_1 = \min\{m, \rho\}.$$

Enough simplifications occur in degenerate cases to justify separate programs for

$$R_C(x, y) = R_F(x, y, y) \quad \text{and} \quad R_D(x, y, z) = R_J(x, y, z, z). \tag{3}$$

Received 21 September 1979, revised 17 March 1980, accepted 9 April 1980.

This work was supported by the U.S. Department of Energy, Division of Basic Energy Sciences, under Contract W-7405-Eng-82.
This work was presented at the 1979 SIAM Spring Meeting, Toronto, Ont., Canada, June 12, 1979.
Authors' address: Department of Mathematics, Iowa State University, Ames, IA 50011.

The latter is an integral of the second kind, while R_C is an elementary function from which inverse circular functions can be computed if $x < y$ and logarithms or inverse hyperbolic functions if $x > y$ (see [1] for explicit formulas). The subroutine RJ calls RC in each cycle of iteration. All four subroutines proceed by successive applications of the duplication theorem, followed by expansion in Taylor series to fifth order. The programs require little space, convergence is linear but quite fast, serious cancellations do not occur, and accuracy is adjustable.

Legendre's elliptic integrals can be computed from these functions, but enumeration of cases is avoided (because of symmetry) by expressing a given elliptic integral directly in terms of R_F, R_D, and R_J. Compact tables of elliptic integrals in terms of these functions are being prepared (e.g., see [2]).

The codes have been verified by PFORT, but fall short of complete portability because they involve two machine-dependent constants, LOLIM and UPLIM, which define the range of admissible arguments. No argument may be greater than UPLIM. The role of LOLIM is less simple, but in most cases it is a lower bound for the sum of two arguments. For R_C and R_F, which are homogeneous of degree $-\frac{1}{2}$, LOLIM and UPLIM differ from the machine minimum and maximum, respectively, by a factor of only 5. For R_D and R_J, which are homogeneous of degree $-\frac{3}{2}$, the limits are inevitably much more restrictive. Values of LOLIM and UPLIM are suggested in the program comments for several types of computers, and formulas are given for use with other machines. A larger (smaller) value of LOLIM (UPLIM) may be used with no ill effects, except that the range of valid arguments will be narrowed. Since the subroutine RJ calls RC, the values of LOLIM and UPLIM for the two programs must be coordinated as specified in the comments.

Our codes have been modified by Dr. J.L. Schonfelder for inclusion in a future edition of the NAG library, and in the case of RC and RF his versions admit all machine-representable arguments. He uses LOLIM and UPLIM to define various regions in which different scaling procedures are used. This entails a significant elaboration of the codes, but will be valuable for users who need to work near the fringes of machine representability.

The accuracy of each program is controlled by a variable called ERRTOL, which determines the maximum error committed in truncating a Taylor series at terms of fifth order. If lower precision is sufficient, a larger value of ERRTOL will speed up the program by requiring fewer iterations of the duplication theorem before computing the Taylor series. Decreasing ERRTOL by a factor of 10 yields six more decimal digits of accuracy at the expense of one or two more iterations. More precisely, decreasing ERRTOL by a factor of 4 requires one more iteration and reduces the error ultimately by a factor of $4^6 = 4096$. Since required accuracy may vary from one run to another, ERRTOL is treated as an input variable. (Its value is set automatically when RC is called by RJ.) On the other hand, LOLIM and UPLIM are set within each subroutine because they will usually be fixed once and for all at a given installation. An exception to the last statement occurs because UPLIM in the case of RD depends on ERRTOL (essentially because the number of iterations depends on ERRTOL, and underflow in computing SIGMA becomes more likely in successive iterations as POWER4 decreases). For nearly all purposes ERRTOL may be replaced by 10^{-5} in the formula for UPLIM, but users who need to admit very large arguments or who want 30 decimal digits of accuracy without losing robustness should recompute UPLIM from the value actually used for ERRTOL.

The programs have been tested for robustness on an ITEL AS/6 computer by running them at extreme points of the region of valid arguments (such as $X = 0$ and $Y =$ UPLIM in the case of RC). In order to achieve robustness the Taylor series given in [1] had to be rearranged in terms of elementary symmetric functions because underflows occurred in computing the power sums $s_n^{(m)}$. By means of [1, e.g., (5.10)] the series in [1, eq. (2.5)] becomes

$$1 - \tfrac{1}{10}E_2 + \tfrac{1}{14}E_3 + \tfrac{1}{24}E_2^2 - \tfrac{3}{44}E_2E_3 + r_n, \tag{4}$$

where E_r is the elementary symmetric function (see [1, Appendix]) of degree r in X_n, Y_n, Z_n. Similarly, the series in [1, (2.19) and (2.27)] become

$$1 - \tfrac{3}{14}E_2 + \tfrac{1}{6}E_3 - \tfrac{3}{22}E_4 + \tfrac{9}{88}E_2^2 + \tfrac{3}{26}E_5 - \tfrac{9}{52}E_2E_3 + r_n, \tag{5}$$

where E_r is an elementary symmetric function of X_n, Y_n, Z_n, P_n, P_n for the case of [1, (2.19)] and X_n, Y_n, Z_n, Z_n, Z_n for [1, (2.27)]. In all cases, $E_1 = 0$.

Some check values are

$$\begin{aligned} R_C(0, \tfrac{1}{4}) &= R_C(\tfrac{1}{16}, \tfrac{1}{8}) = \pi, \\ R_C(\tfrac{9}{4}, 2) &= \log_e 2, \\ R_F(0, 1, 2) &= A = 1.31102\ 87771\ 46059\ 90523\ 24198, \\ R_D(0, 2, 1) &= 3B = 3\pi/4A = 1.79721\ 03521\ 03388\ 31115\ 98837, \\ R_J(2, 3, 4, 5) &= 0.14297\ 57966\ 71567\ 53833\ 23308. \end{aligned} \tag{6}$$

The lemniscate constants A and B are given to 50D in [3]. Consistency checks are provided in the program comments. Values of $(\log x)/(x - 1)$ and $\arctan x$ computed from library routines for the AS/6 computer agree well with those computed from RC, usually to within four units in the last (sixteenth) decimal place. There is similar agreement between RF and the FUNPACK routine for the complete elliptic integral $K(k)$. Because of homogeneity a substantially wider range of values of k^2 can be used with RF than with FUNPACK.

ACKNOWLEDGMENTS

We are grateful for helpful comments and suggestions from J.L. Schonfelder and Edward W. Ng. An important misprint was pointed out by Wayne Fullerton. We owe an exceptional debt to Robert L. Pexton for his interest, encouragement, and unselfish help. He ran many tests on a CDC 7600, and his excellent suggestions and advice have made the programs far better than they would otherwise have been.

REFERENCES

1. CARLSON, B.C. Computing elliptic integrals by duplication. *Numer. Math. 33* (1979), 1–16.
2. CARLSON, B.C. Elliptic integrals of the first kind. *SIAM J. Math. Anal. 8* (1977), 231–242.
3. LEHMER, D.H. The lemniscate constant. *Math. Comput. 3* (1949), 550–551.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service.]

```
C            **********************************************************
C
      DOUBLE PRECISION FUNCTION RC(X, Y, ERRTOL, IERR)
C
C          THIS FUNCTION SUBROUTINE COMPUTES THE ELEMENTARY INTEGRAL
C          RC(X,Y) = INTEGRAL FROM ZERO TO INFINITY OF
C
C                                  -1/2     -1
C                        (1/2)(T+X)     (T+Y)   DT,
C
C          WHERE X IS NONNEGATIVE AND Y IS POSITIVE.  THE DUPLICATION
C          THEOREM IS ITERATED UNTIL THE VARIABLES ARE NEARLY EQUAL,
C          AND THE FUNCTION IS THEN EXPANDED IN TAYLOR SERIES TO FIFTH
C          ORDER.  LOGARITHMIC, INVERSE CIRCULAR, AND INVERSE HYPER-
C          BOLIC FUNCTIONS CAN BE EXPRESSED IN TERMS OF RC.  REFERENCE:
C          B. C. CARLSON, COMPUTING ELLIPTIC INTEGRALS BY DUPLICATION,
C          NUMER. MATH. 33 (1979), 1-16.  CODED BY B. C. CARLSON AND
C          ELAINE M. NOTIS, AMES LABORATORY-DOE, IOWA STATE UNIVERSITY,
C          AMES, IOWA 50011.  MARCH 1, 1980.
C
C          CHECK BY ADDITION THEOREM: RC(X,X+Z) + RC(Y,Y+Z) = RC(0,Z),
C          WHERE X, Y, AND Z ARE POSITIVE AND  X * Y = Z * Z.
```

```
C
      INTEGER IERR, PRINTR
      DOUBLE PRECISION C1, C2, ERRTOL, LAMDA, LOLIM
      DOUBLE PRECISION MU, S, SN, UPLIM, X, XN, Y, YN
C          INTRINSIC FUNCTIONS USED: DABS, DMAX1, DSQRT
C
C          PRINTR IS THE UNIT NUMBER OF THE PRINTER.
      DATA PRINTR /6/
C
C          LOLIM AND UPLIM DETERMINE THE RANGE OF VALID ARGUMENTS.
C          LOLIM IS NOT LESS THAN THE MACHINE MINIMUM MULTIPLIED BY 5.
C          UPLIM IS NOT GREATER THAN THE MACHINE MAXIMUM DIVIDED BY 5.
C
C          ACCEPTABLE VALUES FOR:    LOLIM       UPLIM
C          IBM 360/370 SERIES    :   3.D-78      1.D+75
C          CDC 6000/7000 SERIES  :   1.D-292     1.D+321
C          UNIVAC 1100 SERIES    :   1.D-307     1.D+307
C
C          WARNING: IF THIS PROGRAM IS CONVERTED TO SINGLE PRECISION,
C          THE VALUES FOR THE UNIVAC 1100 SERIES SHOULD BE CHANGED TO
C          LOLIM = 1.E-37 AND UPLIM = 1.E+37 BECAUSE THE MACHINE
C          EXTREMA CHANGE WITH THE PRECISION.
C
      DATA LOLIM /3.D-78/, UPLIM /1.D+75/
C
C          ON INPUT:
C
C          X AND Y ARE THE VARIABLES IN THE INTEGRAL RC(X,Y).
C
C          ERRTOL IS SET TO THE DESIRED ERROR TOLERANCE.
C          RELATIVE ERROR DUE TO TRUNCATION IS LESS THAN
C          16 * ERRTOL ** 6 / (1 - 2 * ERRTOL).
C
C          SAMPLE CHOICES:   ERRTOL    RELATIVE TRUNCATION
C                                      ERROR LESS THAN
C                            1.D-3      2.D-17
C                            3.D-3      2.D-14
C                            1.D-2      2.D-11
C                            3.D-2      2.D-8
C                            1.D-1      2.D-5
C
C          ON OUTPUT:
C
C          X, Y, AND ERRTOL ARE UNALTERED.
C
C          IERR IS THE RETURN ERROR CODE:
C               IERR = 0 FOR NORMAL COMPLETION OF THE SUBROUTINE,
C               IERR = 1 FOR ABNORMAL TERMINATION.
C
C          **********************************************************
C          WARNING: CHANGES IN THE PROGRAM MAY IMPROVE SPEED AT THE
C          EXPENSE OF ROBUSTNESS.
C
      IF (X.LT.0.D0.OR.Y.LE.0.D0) GO TO 10
      IF ((X+Y).LT.LOLIM) GO TO 10
      IF (DMAX1(X,Y).LE.UPLIM) GO TO 20
   10 WRITE (PRINTR,99999)
      WRITE (PRINTR,99998) X, Y
      IERR = 1
      GO TO 50
C
   20 IERR = 0
      XN = X
      YN = Y
C
   30 MU = (XN+YN+YN)/3.D0
      SN = (YN+MU)/MU - 2.D0
      IF (DABS(SN).LT.ERRTOL) GO TO 40
      LAMDA = 2.D0*DSQRT(XN)*DSQRT(YN) + YN
      XN = (XN+LAMDA)*0.25D0
      YN = (YN+LAMDA)*0.25D0
      GO TO 30
C
   40 C1 = 1.D0/7.D0
      C2 = 9.D0/22.D0
      S = SN*SN*(0.3D0+SN*(C1+SN*(0.375D0+SN*C2)))
      RC = (1.D0+S)/DSQRT(MU)
```

```
C
   50 RETURN
99999 FORMAT (1HO, 42H*** ERROR - INVALID ARGUMENTS PASSED TO RC)
99998 FORMAT (1H , 4HX = , D23.16, 4X, 4HY = , D23.16)
      END


C           ***********************************************************
C
      DOUBLE PRECISION FUNCTION RF(X,Y,Z,ERRTOL,IERR)
C
C           THIS FUNCTION SUBROUTINE COMPUTES THE INCOMPLETE ELLIPTIC
C           INTEGRAL OF THE FIRST KIND,
C           RF(X,Y,Z) = INTEGRAL FROM ZERO TO INFINITY OF
C
C                                    -1/2     -1/2     -1/2
C                        (1/2)(T+X)     (T+Y)     (T+Z)     DT,
C
C           WHERE X, Y, AND Z ARE NONNEGATIVE AND AT MOST ONE OF THEM
C           IS ZERO.  IF ONE OF THEM IS ZERO, THE INTEGRAL IS COMPLETE.


C           ***********************************************************
C
      DOUBLE PRECISION FUNCTION RD(X,Y,Z,ERRTOL,IERR)
C
C           THIS FUNCTION SUBROUTINE COMPUTES AN INCOMPLETE ELLIPTIC
C           INTEGRAL OF THE SECOND KIND,
C           RD(X,Y,Z) = INTEGRAL FROM ZERO TO INFINITY OF
C
C                                    -1/2     -1/2     -3/2
C                        (3/2)(T+X)     (T+Y)     (T+Z)     DT,
C
C           WHERE X AND Y ARE NONNEGATIVE, X + Y IS POSITIVE, AND Z IS
C           POSITIVE.  IF X OR Y IS ZERO, THE INTEGRAL IS COMPLETE.


C           ***********************************************************
C
      DOUBLE PRECISION FUNCTION RJ(X,Y,Z,P,ERRTOL,IERR)
C
C           THIS FUNCTION SUBROUTINE COMPUTES AN INCOMPLETE ELLIPTIC
C           INTEGRAL OF THE THIRD KIND,
C           RJ(X,Y,Z,P) = INTEGRAL FROM ZERO TO INFINITY OF
C
C                                      -1/2     -1/2     -1/2     -1
C                          (3/2)(T+X)     (T+Y)     (T+Z)     (T+P)  DT,
C
C           WHERE X, Y, AND Z ARE NONNEGATIVE, AT MOST ONE OF THEM IS
C           ZERO, AND P IS POSITIVE.  IF X OR Y OR Z IS ZERO, THE
C           INTEGRAL IS COMPLETE.
```

ALGORITHM 578
Solution of Real Linear Equations in a Paged Virtual Store [F4]

J. J. DU CROZ and S. M. NUGENT
NAG Central Office, England
J. K. REID
AERE Harwell, England
and
D. B. TAYLOR
University of Edinburgh, Scotland

Key Words and Phrases: Gaussian elimination, paged virtual store
CR Categories: 5.14
Language: FORTRAN

DESCRIPTION

BLCFAC is a routine for performing Gaussian elimination with partial pivoting on a real square matrix A, with the operations on blocks of consecutive columns grouped together to minimize the number of page swaps on a machine with a paged virtual store, as described in [1]. Given that A has been factorized by BLCFAC, the routine BLCSOL will solve either of the systems

$$AX = B$$

or

$$A^{T}X = B$$

with multiple right-hand sides. The operations on blocks of consecutive right-hand sides are grouped together for the same reason. BLCFAC and BLCSOL are essentially the same as the routines F01BTF and F04AYF in the NAG FORTRAN Library.

The routines need two environmental parameters, which are supplied in the functions SRELPR and NSACT. SRELPR is the relative precision, as defined in [2]; NSACT is the target active set size in real storage units. The choice of a suitable value, which depends on the particular machine, is discussed in Section 4 of [1]. The value zero is suitable on nonpaged machines.

Received 16 October 1980; revised 9 March 1981; accepted 30 April 1981

Authors' addresses: J.J. Du Croz and S.M. Nugent, NAG Central Office, 7 Banbury Road, Oxford, Oxon., England; J.K. Reid, Computer Science and Systems Division, Building 8.9, AERE Harwell, Didcot, Oxon., England; D.B. Taylor, Regional Computing Centre, University of Edinburgh, Edinburgh, Scotland.

DOCUMENTATION

The routines are fully documented by embedded comments.

EXAMPLE

To solve the set of linear equations $Ax = b$, where

$$A = \begin{bmatrix} 3 & -6 & 12 & 3 \\ -1 & 4 & -10 & -1 \\ 4 & -13 & 30 & 2 \\ 2 & -11 & 30 & 7 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} -3 \\ 1 \\ -9 \\ 9 \end{bmatrix}$$

```
      INTEGER I, IA, IB, IFAIL, IR, J, N, NIN, NOUT
      LOGICAL TRANS
      REAL A(5, 5), B(5), P(5)
      DATA NIN /5/, NOUT /6/
      N = 4
      READ (NIN, 99999) ((A(I, J), J = 1, N), I = 1, N)
      IA = 5
      IFAIL = 1
C     GAUSSIAN ELIMINATION
      CALL BLCFAC(N, A, IA, P, IFAIL)
      IF (IFAIL.EQ.0) GO TO 20
      WRITE (NOUT, 99998) IFAIL
      STOP
   20 IR = 1
      TRANS = .FALSE.
      READ (NIN, 99999) (B(I), I = 1, N)
      IB = 5
C     APPROXIMATE SOLUTION OF LINEAR EQUATIONS
      CALL BLCSOL(N, IR, TRANS, A, IA, P, B, IB, IFAIL)
      WRITE (NOUT, 99997) (B(I), I = 1, N)
      STOP
99999 FORMAT (4F5.0)
99998 FORMAT (25H0ERROR IN BLCFAC IFAIL = , I2)
99997 FORMAT (10H0SOLUTIONS/(1H , F4.1))
      END
```

The following are suitable data:

```
 3   -6   12   3
-1    4  -10  -1
 4  -13   30   2
 2  -11   30   7
-3    1   -9   9
```

and the following results should be obtained:

```
SOLUTIONS
-1.0
 3.0
 1.0
 2.0
```

REFERENCES

1. Du Croz, J.J., Nugent, S.M., Reid, J.K., and Taylor, D.B. Solving large full sets of linear equations in a paged virtual store. *ACM Trans. Math. Softw. 7*, 4 (Dec. 1981), 527–536.
2. Ford, B. Parameterization of the environment for transportable numerical software. *ACM Trans. Math. Softw. 4*, 2 (June 1978), 100–103.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service.]

```
      SUBROUTINE BLCFAC(N, A, IA, P, IERR)                                BLF  1Ø
C                                                                         BLF  2Ø
C     BLCFAC FACTORIZES A REAL MATRIX BY GAUSSIAN ELIMINATION WITH        BLF  3Ø
C     PARTIAL PIVOTING .  IT IS ORGANIZED SO THAT BLOCKS OF               BLF  4Ø
C     CONSECUTIVE COLUMNS ARE TREATED TOGETHER FOR EFFICIENCY ON A        BLF  5Ø
C     PAGED MACHINE.                                                      BLF  6Ø
C                                                                         BLF  7Ø
C     THE ROUTINE FACTORIZES THE INPUT MATRIX A AS P * L * U,             BLF  8Ø
C     WHERE P IS A PERMUTATION MATRIX, L IS A LOWER TRIANGULAR            BLF  9Ø
C     MATRIX , AND U IS AN UPPER TRIANGULAR MATRIX WITH UNIT              BLF 1ØØ
C     DIAGONAL ELEMENTS.  WHEN CHOOSING A PIVOT, THE ROUTINE              BLF 11Ø
C     IMPLICITLY SCALES THE ROWS TO HAVE LARGEST ELEMENT 1.Ø.  THE        BLF 12Ø
C     ROUTINE TESTS THE SIZE OF THE IMPLICITLY SCALED PIVOT ELEMENT       BLF 13Ø
C     TO CHECK FOR APPROXIMATE SINGULARITY.                               BLF 14Ø
C                                                                         BLF 15Ø
C     PARAMETERS-                                                         BLF 16Ø
C                                                                         BLF 17Ø
C     N       INTEGER                                                     BLF 18Ø
C             ON ENTRY, N SPECIFIES THE ORDER OF THE MATRIX A             BLF 19Ø
C             (N.GT.Ø).  UNCHANGED ON EXIT.                               BLF 2ØØ
C                                                                         BLF 21Ø
C     A       REAL ARRAY OF DIMENSION (IA,N)                              BLF 22Ø
C             BEFORE ENTRY, A(I,J) MUST BE SET TO THE (I,J)TH ELEMENT     BLF 23Ø
C             OF THE MATRIX A, FOR I = 1,...,N AND J = 1,...,N.           BLF 24Ø
C             ON SUCCESSFUL EXIT, THE STRICT UPPER TRIANGLE OF A IS       BLF 25Ø
C             OVERWRITTEN BY THE CORRESPONDING ELEMENTS OF U (THE         BLF 26Ø
C             UNIT DIAGONAL ELEMENTS OF U ARE NOT STORED) AND THE         BLF 27Ø
C             LOWER TRIANGLE OF A IS OVERWRITTEN BY THE MULTIPLIERS       BLF 28Ø
C             USED IN THE GAUSSIAN ELIMINATION (IT DOES NOT CONTAIN       BLF 29Ø
C             THE MATRIX L AS SUCH - THE SUBDIAGONAL ELEMENTS ARE         BLF 3ØØ
C             PERMUTED).                                                  BLF 31Ø
C                                                                         BLF 32Ø
C     IA      INTEGER                                                     BLF 33Ø
C             ON ENTRY, IA SPECIFIES THE FIRST DIMENSION OF A AS          BLF 34Ø
C             DECLARED IN THE CALLING (SUB)PROGRAM (IA.GE.N).             BLF 35Ø
C             UNCHANGED ON EXIT.                                          BLF 36Ø
C                                                                         BLF 37Ø
C     P       REAL ARRAY OF DIMENSION (N)                                 BLF 38Ø
C             ON SUCCESSFUL EXIT, P(I) CONTAINS THE ROW INDEX OF          BLF 39Ø
C             THE I-TH PIVOT, FOR I = 1,...,N.                            BLF 4ØØ
C                                                                         BLF 41Ø
C     IERR    INTEGER                                                     BLF 42Ø
C             IERR IS AN ERROR INDICATOR.  ON EXIT                        BLF 43Ø
C             IERR = Ø IF NO ERROR HAS OCCURRED                           BLF 44Ø
C             IERR = 1 IF ON ENTRY IA.LT.N                                BLF 45Ø
C             IERR = 2 IF THE MATRIX HAS A ROW CONSISTING ENTIRELY        BLF 46Ø
C                      OF ZEROS                                           BLF 47Ø
C             IERR = 3 IF THE MATRIX IS APPROXIMATELY SINGULAR            BLF 48Ø
C             IERR = 4 IF ON ENTRY N.LT.1                                 BLF 49Ø
C                                                                         BLF 5ØØ
      SUBROUTINE BLCSOL(N, IR, TRANS, A, IA, P, B, IB, IERR)              BLS  1Ø
C                                                                         BLS  2Ø
C     BLCSOL CALCULATES THE APPROXIMATE SOLUTION OF A SET OF REAL         BLS  3Ø
C     LINEAR EQUATIONS WITH MULTIPLE RIGHT HAND SIDES A * X = B,          BLS  4Ø
C     AFTER A OR ITS TRANSPOSE HAS BEEN FACTORIZED BY BLCFAC.             BLS  5Ø
C     IT IS ORGANIZED SO THAT BLOCKS OF CONSECUTIVE RIGHT HAND            BLS  6Ø
C     SIDES ARE TREATED TOGETHER FOR EFFICIENCY ON A PAGED MACHINE.       BLS  7Ø
C                                                                         BLS  8Ø
C     PARAMETERS-                                                         BLS  9Ø
C                                                                         BLS 1ØØ
C     N       INTEGER                                                     BLS 11Ø
C             ON ENTRY, N SPECIFIES THE ORDER OF THE MATRIX A             BLS 12Ø
C             (N.GT.Ø).  UNCHANGED ON EXIT.                               BLS 13Ø
C                                                                         BLS 14Ø
C     IR      INTEGER                                                     BLS 15Ø
C             ON ENTRY, IR SPECIFIES THE NUMBER OF RIGHT HAND SIDES       BLS 16Ø
C             I.E.  THE NUMBER OF COLUMNS OF THE MATRIX B (IR.GT.Ø).      BLS 17Ø
```

```
C            UNCHANGED ON EXIT.                                      BLS  18Ø
C                                                                    BLS  19Ø
C     TRANS  LOGICAL                                                 BLS  2ØØ
C            ON ENTRY, TRANS MUST BE TRUE IF THE TRANSPOSE OF A      BLS  21Ø
C            WAS FACTORIZED BY BLCSOL AND FALSE OTHERWISE.           BLS  22Ø
C            UNCHANGED ON EXIT.                                      BLS  23Ø
C                                                                    BLS  24Ø
C     A      REAL ARRAY OF DIMENSION (IA,N)                          BLS  25Ø
C            BEFORE ENTRY, A MUST CONTAIN THE FACTORIZATION OF THE   BLS  26Ø
C            MATRIX A OR ITS TRANSPOSE AS RETURNED BY THE            BLS  27Ø
C            ROUTINE BLCSOL. UNCHANGED ON EXIT.                      BLS  28Ø
C                                                                    BLS  29Ø
C     IA     INTEGER                                                 BLS  3ØØ
C            ON ENTRY, IA SPECIFIES THE FIRST DIMENSION OF A AS      BLS  31Ø
C            DECLARED IN THE CALLING (SUB)PROGRAM (IA.GE.N).         BLS  32Ø
C            UNCHANGED ON EXIT.                                      BLS  33Ø
C                                                                    BLS  34Ø
C     P      REAL ARRAY OF DIMENSION (N)                             BLS  35Ø
C            BEFORE ENTRY, P MUST CONTAIN THE DETAILS OF THE ROW     BLS  36Ø
C            PERMUTATIONS AS RETURNED BY THE ROUTINE BLCFAC          BLS  37Ø
C                                                                    BLS  38Ø
C     B      REAL ARRAY OF DIMENSION (IB,IR)                         BLS  39Ø
C            BEFORE ENTRY, B MUST CONTAIN THE MATRIX OF RIGHT        BLS  4ØØ
C            HAND SIDES B.                                           BLS  41Ø
C            ON SUCCESSFUL EXIT, B IS OVERWRITTEN BY THE MATRIX      BLS  42Ø
C            OF SOLUTIONS X.                                         BLS  43Ø
C                                                                    BLS  44Ø
C     IB     INTEGER                                                 BLS  45Ø
C            ON ENTRY, IB SPECIFIES THE FIRST DIMENSION OF B AS      BLS  46Ø
C            DECLARED IN THE CALLING (SUB)PROGRAM (IB.GE.N).         BLS  47Ø
C            UNCHANGED ON EXIT.                                      BLS  48Ø
C                                                                    BLS  49Ø
C     IERR   INTEGER                                                 BLS  5ØØ
C            IERR IS AN ERROR INDICATOR.  ON EXIT                    BLS  51Ø
C            IERR = Ø IF NO ERROR HAS OCCURRED                       BLS  52Ø
C            IERR = 1 IF ON ENTRY IA.LT.N                            BLS  53Ø
C            IERR = 2 IF ON ENTRY IR.LT.1                            BLS  54Ø
C            IERR = 3 IF ON ENTRY IB.LT.N                            BLS  55Ø
C            IERR = 4 IF ON ENTRY N.LT.1                             BLS  56Ø
C                                                                    BLS  57Ø
```

ALGORITHM 579
CPSC: Complex Power Series Coefficients [D4]

BENGT FORNBERG
California Institute of Technology

Key Words and Phrases: numerical differentiation, Taylor series coefficients, analytic functions
CR Categories: 5.16
Language: FORTRAN

DESCRIPTION

An algorithm CPSC is presented here. It evaluates numerically the leading coefficients in a power series expansion of an analytic function (or, equivalently, a number of leading derivatives of an analytic function). A detailed description of the theoretical background of the code is given in [1], together with some warnings about cases in which full accuracy may not be reached.

Such cases are

(1) very low-order polynomials (for example, $f(z) = 1 + z$);
(2) functions whose Taylor expansions contain very large isolated terms (for example, $f(z) = 10^6 + (1/(1 - z))$);
(3) certain entire functions (for example, $f(z) = e^z$);
(4) functions whose radius of convergence is limited by a branch point at which the function remains many times differentiable (for example, $f(z) = (1 + z)^{10} \log(1 + z)$ expanded around $z = 0$).

In the case 1, the routine will normally fail to even approximate the correct answer. In cases 2, 3, and 4, problems are normally encountered only if large numbers of coefficients are wanted (e.g., more than 30). The supplied error estimate will, in most of these cases, give correct information about the lowered accuracy.

Inevitably there is a risk that the routine will attempt to evaluate the given function exactly at a singularity. This will happen, for example, for $f(z) = 1/(1 - z)$ expanded around zero if the initial radius is of the form $r = 2^p$, with p any integer. The risk for such exact coincidences in floating point can be minimized by choosing "irregular" initial radii. A practical procedure may be to start each case by the obtained final radius from a previous case.

Received 28 August 1978; revised 6 February 1980; accepted 19 June 1981

This work was supported by Control Data Corporation and by the Department of Energy (Office of Basic Energy Sciences).

Author's address: Department of Applied Mathematics, 217-50 Caltech, Pasadena, CA 91125.

REFERENCES

1. FORNBERG, B. Numerical differentiation of analytic functions. *ACM Trans. Math. Softw. 7*, 4 (Dec. 1981), 512-526.

ALGORITHM

The user must enter the machine accuracy in a data statement. The routine adjusts the calculations accordingly in order to obtain the best possible accuracy. However, since a few decimal places are always lost in this routine, we do not recommend its general use if the machine accuracy is less than 10 significant decimal places. For example, for use on IBM System 370 machines, we recommend double or quadruple precision. The code below is for IBM double precision.

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service.]

```
C CPSC - COMPLEX POWER SERIES COEFFICIENTS
C
C BY BENGT FORNBERG
C
C ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE, DECEMBER 1981
C
      PROGRAM TEST (OUTPUT)
C
C  THIS PROGRAM WITH THE FUNCTION F BELOW USES CPSC TO EVALUATE
C  THE LEADING DERIVATIVES OF F(Z) = CEXP(Z)/(CSIN(Z)**3+CCOS(Z)**3)
C  AT Z = Ø.
      EXTERNAL F
      DIMENSION A(51),IRANGE(4),ER(51)
      COMPLEX A
      DATA IRANGE/6,12,25,51/
      DO 1Ø I=1,4
        IR = IRANGE(I)
        R = 1.
        CALL CPSC(F,(Ø.,Ø.),IR,1,R,A,ER)
        WRITE(6,2Ø) I
        WRITE(6,3Ø) (A(J),J=1,IR)
   1Ø   WRITE(6,4Ø) (ER(J),J=1,IR)
   2Ø FORMAT(/2ØH DERIVATIVES , RANGE,I3)
   3Ø FORMAT(4(1X,E18.1Ø,E9.1))
   4Ø FORMAT(/17H ESTIMATED ERRORS/(1X,16E8.1))
      STOP
      END
      COMPLEX FUNCTION F(Z)
C
C  TEST FUNCTION FOR USE WITH CPSC.
C
      COMPLEX Z
      F = CEXP(Z)/(CSIN(Z)**3+CCOS(Z)**3)
      RETURN
      END


      SUBROUTINE CPSC(F,Z,N,IC,R,RS,ER)
C
C  EVALUATION OF COMPLEX POWER SERIES COEFFICIENTS OR DERIVATIVES.
C
C  *** INPUT PARAMETERS ***
C    F  COMPLEX FUNCTION, OF WHICH THE COEFFICIENTS OR DERIVATIVES
C       ARE SOUGHT. THIS FUNCTION MUST BE DECLARED EXTERNAL IN THE
C       CALLING PROGRAM.
C    Z  COMPLEX POINT AROUND WHICH F SHALL BE EXPANDED OR AT WHICH
C       DERIVATIVES SHALL BE EVALUATED.
C    N  INTEGER, NUMBER OF COEFFICIENTS OR DERIVATIVES WANTED.
C       N MUST BE GE 1 AND LE 51.
C    IC SELECTS BETWEEN POWER SERIES COEFFICIENTS AND DREIVATIVES.
C       IC .EQ. Ø  ROUTINE RETURNS POWER SERIES COEFFICIENTS IN  RS
C       IC .NE. Ø  ROUTINE RETURNS DERIVATIVES IN RS .
C  *** INPUT AND OUTPUT PARAMETER ***
C    R  INITIAL RADIUS USED IN SEARCH FOR OPTIMAL RADIUS. THE RESULTING
```

```
C       RADIUS IS LEFT IN R. THE PROVIDED GUESS MAY BE IN ERROR WITH AT
C       MOST A FACTOR OF 3.E4 .
C  *** OUTPUT PARAMETERS ***
C    RS COMPLEX ARRAY RS(N) CONTAINING THE N FIRST
C       COEFFICIENTS (CORRESPONDING TO THE POWERS Ø TO N-1) OR DERIVA-
C       TIVES (ORDERS Ø TO N-1) .
C    ER REAL ARRAY ER(N) CONTAINING ERROR ESTIMATES FOR THE
C       NUMBERS IN RS(N).
C
      DIMENSION IP(32),A(64),RS(N),ER(N),RT(51,3),FV(6),
     *  IW(7),SC(4),RV(3),C(4),FC(3)
      COMPLEX F,A,V,RS,RT,FV,U,W,T,Z,RV,RQ,S,XK,MULT,CO
C
C  LIST OF THE VARIABLES INITIALIZED IN THE DATA STATEMENT BELOW.
C   EPS  MACHINE ACCURACY. THIS CONSTANT HAS TO BE SUPPLIED BY
C        THE USER. IN THIS LIST, IT IS GIVEN AS 1.E-14 CORRESPONDING
C        TO THE 48 BIT FLOATING POINT MANTISSAS ON CDC CYBER MACHINES.
C   IND  INTEGER FLAG.
C   L2   INTEGER FLAG.
C   IW   2**( Ø , 1 , 2 , 3 , 4 , 5 , 6 ) .
C   IP   PERMUTATION CONSTANTS FOR THE FFT.
C   RV   CONSTANTS FOR THE LAURENT SERIES TEST .
C
      DATA EPS/1.E-14/,IND/Ø/,L2/1/,IW/1,2,4,8,16,32,64/,
     *  IP/64,32,48,16,56,24,4Ø,8,6Ø,28,44,12,52,2Ø,36,4,62,3Ø,46,14,
     *    54,22,38,6,58,26,42,1Ø,5Ø,18,34,2/,
     *  RV/(-.4,.3),(.7,.2),(.Ø2,-.Ø6)/
C
C  STATEMENT FUNCTION FOR MULTIPLICATION OF A COMPLEX NUMBER
C  BY A REAL NUMBER.
C
      MULT(RE,CO) = CMPLX(RE*REAL(CO),RE*AIMAG(CO))
C
C  EVALUATE SOME CONSTANTS THE FIRST TIME THE CODE IS EXECUTED.
C
      IF(IND.EQ.1) GOTO 2Ø
      IND = 1
      SC(1) = .125
      C(1) = EPS**(1./28.)
      EP6 = C(1)**6
      PI = 4.*ATAN(1.)
      FV(1) = (-1.,Ø.)
      FV(2) = (Ø.,-1.)
      R1 = SQRT(.5)
      RA = 1./R1
      FV(3) = CMPLX(R1,-R1)
      DO 1Ø I=2,4
        SC(I) = .5*SC(I-1)
        C(I) = SQRT(C(I-1))
        ANG = PI*SC(I-1)
   1Ø   FV(I+2) = CMPLX(COS(ANG),-SIN(ANG))
   2Ø CONTINUE
C
C  START EXECUTION.
C
      IF(N.GT.51.OR.N.LT.1) GOTO 26Ø
      LF = Ø
      NP = Ø
      M = Ø
      NR = -1
C
C  FIND IF A FFT OVER 8, 16, 32, OR 64 POINTS SHOULD BE USED.
C
      KL = 1
      IF(N.GT.6) KL=2
      IF(N.GT.12) KL=3
      IF(N.GT.25) KL=4
      KM = KL+2
      KN = 7-KM
      IX = IW(KM+1)
      IS = IW(KN)
   3Ø V = CMPLX(R,Ø.)
C
```

```
C  FUNCTION VALUES OF F ARE STORED READY PERMUTATED FOR THE FFT.
C
      DO 4Ø I=IS,32,IS
        IQ = IP(I)
        V = V*FV(KM)
        A(IQ) = F(Z+V)
   4Ø   A(IQ-1) = F(Z-V)
      LN = 2
      JN = 1
C
C  THE LOOP  DO 7Ø ... CONSTITUTES THE FFT.
C
      DO 7Ø L=1,KM
        U = (1.,Ø.)
        W = FV(L)
        DO 6Ø J=1,JN
          DO 5Ø I=J,IX,LN
            IT = I+JN
            T = A(IT)*U
            A(IT) = A(I)-T
   5Ø       A(I) = A(I)+T
   6Ø     U = U*W
        LN = LN+LN
   7Ø   JN = JN+JN
      CX = Ø.
      B = 1.
C
C  TEST ON HOW FAST THE COEFFICIENTS OBTAINED DECREASE.
C
      DO 8Ø I=1,IX
        CT = CABS(A(I))/B
        IF(CT.LT.CX) GOTO 8Ø
        CX = CT
        INR = I
   8Ø   B = B*C(KL)
      IF(M.LE.1) GOTO 1ØØ
C
C  ESTIMATE OF THE ROUNDING ERROR LEVEL FOR THE LAST RADIUS.
C
      ER(1) = CX*EPS
      DO 9Ø I=2,N
   9Ø   ER(I) = ER(I-1)/R
  1ØØ SF = SC(KL)
      DO 11Ø I=1,IX
        A(I) = MULT(SF,A(I))
  11Ø   SF = SF/R
      L1 = L2
      L2 = 1
      IF(INR.GT.IW(KM)) GOTO 15Ø
      IF(LF.EQ.1) GOTO 14Ø
C
C  TEST IF THE SERIES IS A TAYLOR OR A LAURENT SERIES.
C
      SR = Ø.
      SP = Ø.
      DO 13Ø J=1,3
        RQ = MULT(R,RV(J))
        S = A(IX)
        DO 12Ø I=2,IX
          IA = IX+1-I
  12Ø     S = S*RQ+A(IA)
        CP = CABS(S)
        IF(CP.GT.SP) SP=CP
        CM = CABS(S-F(Z+RQ))
  13Ø   IF(CM.GT.SR) SR=CM
      IF(SR.GT.1.E-3*SP) GOTO 15Ø
      LF = 1
  14Ø L2 = -1
C
C  DETERMINATION OF THE NEXT RADIUS TO BE USED.
C
  15Ø IF(NR.GE.Ø) GOTO 16Ø
      FACT = 2.
```

```
      IF(L2.EQ.1) FACT=.5
      L1 = L2
      NR = Ø
  16Ø IF(L1.NE.L2) GOTO 18Ø
      IF(NR.GT.Ø) GOTO 17Ø
      NP = NP+1
      IF(NP-15) 19Ø,19Ø,26Ø
  17Ø FACT = 1./FACT
  18Ø FACT = 1./SQRT(FACT)
      NR = NR+1
  19Ø R = R*FACT
      M = NR-KL-1
      IF(M.LE.Ø) GOTO 3Ø
C
C  THE RESULTS FOR THE LAST THREE RADII ARE STORED.
C
      DO 2ØØ I=1,N
  2ØØ   RT(I,M) = A(I)
      IF(M.EQ.1) GOTO 22Ø
C
C  EXTRAPOLATION.
C
      DO 21Ø I=1,N
        XK = RT(I,M-1)-RT(I,M)
  21Ø   RT(I,M-1) = RT(I,M)-MULT(FC(M-1),XK)
      IF(M.EQ.3) GOTO 23Ø
C
C  CALCULATION OF THE EXTRAPOLATION CONSTANTS.
C
  22Ø FC(M) = 1.5+SIGN(.5,FACT-1.)
      IF(M.EQ.2) FC(M)=FC(M)+RA
      IF(FACT.GT.1.) FC(M)=-FC(M)
      GOTO 3Ø
  23Ø FC(3) = FC(1)*FC(2)/(FC(1)+FC(2)+1.)
C
C  FINAL EXTRAPOLATION AND ERROR ESTIMATE.
C
      DO 24Ø I=1,N
        XK = RT(I,1)-RT(I,2)
        ER(I) = ER(I)+EP6*CABS(XK)
  24Ø   RS(I) = RT(I,2)-MULT(FC(3),XK)
C
C  MULTIPLY POWER SERIES COEFFICIENTS AND ERROR ESTIMATE BY FACTORIALS
C  IF DERIVATIVES WANTED.
C
      IF(IC.EQ.Ø) RETURN
      FAC = Ø.
      FACT = 1.
      DO 25Ø I=1,N
        RS(I) = MULT(FACT,RS(I))
        ER(I) = FACT*ER(I)
        FAC = FAC+1.
  25Ø   FACT = FACT*FAC
      RETURN
C
C  ERROR RETURN.
C
  26Ø DO 27Ø I=1,N
        RS(I) = (Ø.,Ø.)
  27Ø   ER(I) = 1.E1Ø
      RETURN
      END
```

ALGORITHM 580
QRUP: A Set of FORTRAN Routines for Updating QR Factorizations [F5]

A. BUCKLEY
Concordia University, Canada

Key Words and Phrases: matrix factorization, orthogonalization
CR Categories: 5.14
Language: FORTRAN

DESCRIPTION

This algorithm is a translation into FORTRAN of the ALGOL routines published by Daniel et al. [2]. Given an $m \times n$ matrix A with $m \geq n$, these routines compute a QR factorization of A based on the Gram–Schmidt orthogonalization process, and they provide for updating the factorization when rows or columns are added to or deleted from A. Also the factors may be updated when A is modified by the addition or subtraction of a rank one matrix.

The translation is in most instances straightforward, but there are some points that need to be noted. Most of these are discussed in detail in the descriptive routine DESCRB, which is part of the algorithm.

However, a few comments belong here. Notice that DESCRB contains both a complete description of the routines and a fairly lengthy set of tests to ensure the accuracy of the translation. The tests basically consist of an initial factorization of a 7×7 Hilbert matrix, followed by random deletion of rows and columns of the matrix until it is just 1×1, after which the rows and columns are, in a random order, reinserted to recreate the original matrix. Rank one matrices are introduced (then deleted) at several points as well. At each step, the factorization is updated and its accuracy is checked against the matrix being factored. A copy of the output from an execution of the tests follows the algorithm. The routines have also been tested through their use in an implementation of the minimization algorithm described in [1].

The translation is written in a portable subset of American National Standard FORTRAN, 1966, and has been thoroughly checked by the PFORT verifier. Except for some minor points noted in the routine DESCRB, the code is machine independent.

The routines are written simultaneously in single and double precision. As listed, compilation will give a single-precision version. It is straightforward,

Received 15 August 1979; revised 22 August 1980; accepted 1 April 1981

This work was supported by NSERC Grant A8962.

Author's address: Mathematics Department, Layola Campus, Concordia University, 7141 Sherbrooke Street W, Montreal, Que., H4B 1R6, Canada.

however, to obtain a double-precision version: just follow the simple instructions in DESCRB to interchange a few well-marked statements.

Notice that the basic linear algebra operations, such as taking norms or computing inner products, are done via calls to the BLAS described by Lawson et al. [3]. These routines are available through the International Mathematical and Statistical Libraries, Inc.,[1] or through SIAM.[2]

ACKNOWLEDGMENT

The author wishes to indicate his appreciation of the support received from NSERC, Canada.

REFERENCES

1. BUCKLEY, A.G. Constrained minimization using Powell's conjugacy approach. *SIAM J. Numer. Anal. 13* (Oct. 1976), 520–535.
2. DANIEL, J., GRAGG, W.B., KAUFMAN, L., AND STEWART, G.W. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Math. Comput. 30* (Oct. 1976), 772–795.
3. LAWSON, C.L., HANSON, R.J., KINCAID, D.R., AND KROGH, F.T. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Softw. 5*, 3 (Sept. 1979), 308–323.

[1] NBC Building, 7500 Bellaire, Houston, TX 77036.
[2] 117 South 17 St., Philadelphia, PA 19103.

ALGORITHM

[A complete listing of the algorithm is available from the ACM Algorithms Distribution Service.]

REMARK ON ALGORITHM 580

QRUP: A Set of FORTRAN Routines for Updating QR Factorizations [A. Buckley, *ACM Trans. Math. Softw. 7*, 4 (Dec. 1981), 548–549]

A. Buckley [Received 20 May 1982; accepted 20 May 1982]
Department of Management Information Systems, University of Arizona, Tucson, AZ 85721.

In two subprograms the functions **DNRM2( )** and **DDOT( )** are not declared to be **DOUBLE PRECISION** as they should be.

In program unit **DESCRB( )**, sequence numbers **QRUP 139** and **QRUP 140** should be modified to read:

```
      REAL                A, ASS, SDOT, COL, GLRMAX, OMEGA
C!!!! DOUBLE PRECISION    A, ASS, DDOT, COL, GLRMAX, OMEGA
```

In program unit **ORTCOL( )**, sequence numbers **QRUP 762** and **QRUP 763** should be modified to read:

```
      REAL                SDOT, SNRM2, OMEGA, ONE, ONENEG, Q, RHO
C!!!! DOUBLE PRECISION    DDOT, DNRM2, OMEGA, ONE, ONENEG, Q, RHO
```

ALGORITHM 581
An Improved Algorithm for Computing the Singular Value Decomposition [F1]

TONY F. CHAN
Department of Computer Science, Yale University

Categories and Subject Descriptors: D.3.2 [**Programming Languages**]: Language Classifications—*FORTRAN*; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*pseudoinverses*; G.1.6 [**Numerical Analysis**]: Optimizations—*least squares methods*

General Terms: Algorithms

Additional Key Words and Phrases: singular value decomposition

DESCRIPTION

The set of FORTRAN subroutines given here is an implementation of the algorithm [1] for computing the Singular Value Decomposition (SVD) of a general m by n rectangular matrix A defined as

$$A = UWV^{T},$$

where U is an $m \times \min(m,n)$ matrix containing the left singular vectors, W is a diagonal matrix of size $\min(m, n)$ containing the singular values, and V is an $n \times \min(m, n)$ matrix containing the right singular vectors. Note that m is allowed to be greater than or less than n. For ease of presentation, we assume m to be greater than or equal to n in the following discussion.

The algorithm is an improvement of the Golub–Reinsch algorithm [4], which is implemented in subroutines SVD and MINFIT in EISPACK [3] and in subroutine SSVDC in LINPACK [2]. It should be more efficient than the Golub–Reinsch algorithm when m is approximately larger than $2n$, as is the case in many least squares applications.

The algorithm has a hybrid nature. When m is about equal to n, the Golub–Reinsch algorithm is employed. When the ratio m/n is larger than a threshold value, which is determined by detailed operation counts [1], the improved algorithm is used.

The improved algorithm first computes the QR factorization of A using Householder transformations, and then uses the Golub–Reinsch algorithm on R. A further improvement over the Golub–Reinsch algorithm is when the left singular vectors are to be accumulated and saved. Here, instead of accumulating the Givens transformations (in the second phase of the algorithm where the singular values of the bidiagonal matrix are computed) on the $m \times n$ matrix containing the left singular vectors, we accumulate them on a temporary $n \times n$ matrix. This requires a small overhead in storage of an $n \times n$ matrix (small compared with $m \times n$) but offers big savings in time.

Received 7 October 1976; revised 25 January 1982; accepted 29 January 1982
This work was supported by NSF Grant DCR 75-13497 and NASA Ames Contract NCA 2-OR745-520. The computing time was provided by the Stanford Linear Accelerator Center (SLAC).
Author's address: Department of Computer Science, Yale University, 10 Hillhouse Avenue, P.O. Box 2158 Yale Station, New Haven CT 06520.

An additional feature of the new algorithm is that it can accumulate all the left orthogonal transformations on a number of given vectors, which can then be used in computing least squares solutions. In this fashion, it is similar to the EISPACK routine MINFIT.

There are three main routines in the package:

HYBSVD: This is the main routine which implements the hybrid algorithm.
MGNSVD: This performs the same thing as HYBSVD, except that it assumes m .ge. n.
GRSVD: This is a slightly modified version of routine SVD in EISPACK which implements the Golub-Reinsch algorithm.

Besides, there are two utility routines:

SSWAP: BLAS routine for swapping two vectors.
SRELPR: Routine for computing the machine relative precision.

These five routines must be used together. They have been tested extensively on the IBM 370/168, 360/91 at the Stanford Linear Accelerator Center, and on the DEC 2060 in the Computer Science Department at Yale. They produce results that agree (up to machine precision) with those produced by SVD, MINFIT, and SSVDC. They have been verified by PFORT verifier [5] for portability.

REFERENCES

1. CHAN, T.F. An improved algorithm for computing the Singular Value Decomposition. *ACM Trans, Math. Softw. 8*, 1 (Mar. 1982), 72–83.
2. DONGARRA, J.J., BUNCH, J.R., MOLER, C.B., AND STEWART, G.W. *LINPACK User's Guide.* SIAM, Philadelphia, 1979.
3. GARBOW, B.S., ET AL. *Matrix Eigensystem Routines—EISPACK Guide Extension*, Lecture Notes in Computer Science Series, No. 51. Springer-Verlag, New York, 1977.
4. GOLUB, G.H., AND REINSCH, C. Singular Value Decomposition and least squares solutions. In *Handbook for Automatic Computation*, II, *Linear Algebra*, J.H. Wilkinson and C. Reinsch (Eds.), Springer-Verlag, New York, 1971.
5. RYDER, B.G. The PFORT verifier. In *Softw. Pract. Exper. 4* (1974) 359–377.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service (see page 91 for order form).]

```
      SUBROUTINE HYBSVD(NA, NU, NV, NZ, NB, M, N, A, W, MATU, U, MATV,  HYB   1Ø
     * V, Z, B, IRHS, IERR, RV1)                                       HYB   2Ø
      INTEGER NA, NU, NV, NZ, M, N, IRHS, IERR, MINØ                   HYB   3Ø
      REAL A(NA,1), W(1), U(NU,1), V(NV,1), Z(NZ,1), B(NB,IRHS), RV1(1) HYB   4Ø
      LOGICAL MATU, MATV                                               HYB   5Ø
C                                                                      HYB   6Ø
C     THIS ROUTINE IS A MODIFICATION OF THE GOLUB-REINSCH PROCEDURE (1) HYB   7Ø
C                                                          T           HYB   8Ø
C     FOR COMPUTING THE SINGULAR VALUE DECOMPOSITION A = UWV  OF A     HYB   9Ø
C     REAL M BY N RECTANGULAR MATRIX. U IS M BY MIN(M,N) CONTAINING    HYB  1ØØ
C     THE LEFT SINGULAR VECTORS, W IS A MIN(M,N) BY MIN(M,N) DIAGONAL  HYB  11Ø
C     MATRIX CONTAINING THE SINGULAR VALUES, AND V IS N BY MIN(M,N)    HYB  12Ø
C     CONTAINING THE RIGHT SINGULAR VECTORS.                           HYB  13Ø
C                                                                      HYB  14Ø
C     THE ALGORITHM IMPLEMENTED IN THIS                                HYB  15Ø
C     ROUTINE HAS A HYBRID NATURE.  WHEN M IS APPROXIMATELY EQUAL TO N, HYB  16Ø
C     THE GOLUB-REINSCH ALGORITHM IS USED, BUT WHEN EITHER OF THE RATIOSHYB  17Ø
C     M/N OR N/M IS GREATER THAN ABOUT 2,                              HYB  18Ø
C     A MODIFIED VERSION OF THE GOLUB-REINSCH                          HYB  19Ø
C     ALGORITHM IS USED.  THIS MODIFIED ALGORITHM FIRST TRANSFORMS A   HYB  2ØØ
C                                                              T       HYB  21Ø
C     INTO UPPER TRIANGULAR FORM BY HOUSEHOLDER TRANSFORMATIONS L      HYB  22Ø
C     AND THEN USES THE GOLUB-REINSCH ALGORITHM TO FIND THE SINGULAR   HYB  23Ø
C     VALUE DECOMPOSITION OF THE RESULTING UPPER TRIANGULAR MATRIX R.  HYB  24Ø
C     WHEN U IS NEEDED EXPLICITLY IN THE CASE M.GE.N (OR V IN THE CASE HYB  25Ø
C     M.LT.N), AN EXTRA ARRAY Z (OF SIZE AT LEAST                      HYB  26Ø
C     MIN(M,N)**2) IS NEEDED, BUT OTHERWISE Z IS NOT REFERENCED        HYB  27Ø
C     AND NO EXTRA STORAGE IS REQUIRED.  THIS HYBRID METHOD            HYB  28Ø
```

```
C     SHOULD BE MORE EFFICIENT THAN THE GOLUB-REINSCH ALGORITHM WHEN    HYB  29Ø
C     M/N OR N/M IS LARGE.  FOR DETAILS, SEE (2).                      HYB  3ØØ
C                                                                      HYB  31Ø
C     WHEN M .GE. N,                                                   HYB  32Ø
C     HYBSVD CAN ALSO BE USED TO COMPUTE THE MINIMAL LENGTH LEAST      HYB  33Ø
C     SQUARES SOLUTION TO THE OVERDETERMINED LINEAR SYSTEM A*X=B.       HYB  34Ø
C     IF M .LT. N (I.E. FOR UNDERDETERMINED SYSTEMS), THE RHS B        HYB  35Ø
C     IS NOT PROCESSED.                                                HYB  36Ø
C                                                                      HYB  37Ø
C     NOTICE THAT THE SINGULAR VALUE DECOMPOSITION OF A MATRIX         HYB  38Ø
C     IS UNIQUE ONLY UP TO THE SIGN OF THE CORRESPONDING COLUMNS       HYB  39Ø
C     OF U AND V.                                                      HYB  4ØØ
C                                                                      HYB  41Ø
C     THIS ROUTINE HAS BEEN CHECKED BY THE PFORT VERIFIER (3) FOR      HYB  42Ø
C     ADHERENCE TO A LARGE, CAREFULLY DEFINED, PORTABLE SUBSET OF      HYB  43Ø
C     AMERICAN NATIONAL STANDARD FORTRAN CALLED PFORT.                 HYB  44Ø
C                                                                      HYB  45Ø
C     REFERENCES:                                                      HYB  46Ø
C                                                                      HYB  47Ø
C     (1) GOLUB,G.H. AND REINSCH,C. (197Ø) 'SINGULAR VALUE             HYB  48Ø
C         DECOMPOSITION AND LEAST SQUARES SOLUTIONS,'                  HYB  49Ø
C         NUMER. MATH. 14,4Ø3-42Ø, 197Ø.                               HYB  5ØØ
C                                                                      HYB  51Ø
C     (2) CHAN,T.F. (1982) 'AN IMPROVED ALGORITHM FOR COMPUTING        HYB  52Ø
C         THE SINGULAR VALUE DECOMPOSITION,' ACM TOMS, VOL.8,          HYB  53Ø
C         NO. 1, MARCH, 1982.                                          HYB  54Ø
C                                                                      HYB  55Ø
C     (3) RYDER,B.G. (1974) 'THE PFORT VERIFIER,' SOFTWARE -           HYB  56Ø
C         PRACTICE AND EXPERIENCE, VOL.4, 359-377, 1974.               HYB  57Ø
C                                                                      HYB  58Ø
C     ON INPUT:                                                        HYB  59Ø
C                                                                      HYB  6ØØ
C        NA MUST BE SET TO THE ROW DIMENSION OF THE TWO-DIMENSIONAL    HYB  61Ø
C          ARRAY PARAMETER A AS DECLARED IN THE CALLING PROGRAM        HYB  62Ø
C          DIMENSION STATEMENT.  NOTE THAT NA MUST BE AT LEAST         HYB  63Ø
C          AS LARGE AS M.                                              HYB  64Ø
C                                                                      HYB  65Ø
C        NU MUST BE SET TO THE ROW DIMENSION OF THE TWO-DIMENSIONAL    HYB  66Ø
C          ARRAY U AS DECLARED IN THE CALLING PROGRAM DIMENSION        HYB  67Ø
C          STATEMENT. NU MUST BE AT LEAST AS LARGE AS M.               HYB  68Ø
C                                                                      HYB  69Ø
C        NV MUST BE SET TO THE ROW DIMENSION OF THE TWO-DIMENSIONAL    HYB  7ØØ
C          ARRAY PARAMETER V AS DECLARED IN THE CALLING PROGRAM        HYB  71Ø
C          DIMENSION STATEMENT. NV MUST BE AT LEAST AS LARGE AS N.     HYB  72Ø
C                                                                      HYB  73Ø
C        NZ MUST BE SET TO THE ROW DIMENSION OF THE TWO-DIMENSIONAL    HYB  74Ø
C          ARRAY PARAMETER Z AS DECLARED IN THE CALLING PROGRAM        HYB  75Ø
C          DIMENSION STATEMENT.  NOTE THAT NZ MUST BE AT LEAST         HYB  76Ø
C          AS LARGE AS MIN(M,N).                                       HYB  77Ø
C                                                                      HYB  78Ø
C        NB MUST BE SET TO THE ROW DIMENSION OF THE TWO-DIMENSIONAL    HYB  79Ø
C          ARRAY PARAMETER B AS DECLARED IN THE CALLING PROGRAM        HYB  8ØØ
C          DIMENSION STATEMENT. NB MUST BE AT LEAST AS LARGE AS M.     HYB  81Ø
C                                                                      HYB  82Ø
C        M IS THE NUMBER OF ROWS OF A (AND U).                         HYB  83Ø
C                                                                      HYB  84Ø
C        N IS THE NUMBER OF COLUMNS OF A (AND NUMBER OF ROWS OF V).    HYB  85Ø
C                                                                      HYB  86Ø
C        A CONTAINS THE RECTANGULAR INPUT MATRIX TO BE DECOMPOSED.     HYB  87Ø
C                                                                      HYB  88Ø
C        B CONTAINS THE IRHS RIGHT-HAND-SIDES OF THE OVERDETERMINED    HYB  89Ø
C         LINEAR SYSTEM A*X=B. IF IRHS .GT. Ø AND M .GE. N,            HYB  9ØØ
C         THEN ON OUTPUT, THE FIRST N COMPONENTS OF THESE IRHS COLUMNS HYB  91Ø
C                   T                                                  HYB  92Ø
C         WILL CONTAIN U B. THUS, TO COMPUTE THE MINIMAL LENGTH LEAST  HYB  93Ø
C                                        +                             HYB  94Ø
C         SQUARES SOLUTION, ONE MUST COMPUTE V*W  TIMES THE COLUMNS OF HYB  95Ø
C                   +                      +                           HYB  96Ø
C         B, WHERE W  IS A DIAGONAL MATRIX, W (I)=Ø IF W(I) IS          HYB  97Ø
C         NEGLIGIBLE, OTHERWISE IS 1/W(I). IF IRHS=Ø OR M.LT.N,        HYB  98Ø
C         B IS NOT REFERENCED.                                         HYB  99Ø
C                                                                      HYB 1ØØØ
C        IRHS IS THE NUMBER OF RIGHT-HAND-SIDES OF THE OVERDETERMINED  HYB 1Ø1Ø
C         SYSTEM A*X=B. IRHS SHOULD BE SET TO ZERO IF ONLY THE SINGULAR HYB 1Ø2Ø
C         VALUE DECOMPOSITION OF A IS DESIRED.                         HYB 1Ø3Ø
C                                                                      HYB 1Ø4Ø
```

```
C        MATU SHOULD BE SET TO .TRUE. IF THE U MATRIX IN THE           HYB 1Ø5Ø
C          DECOMPOSITION IS DESIRED, AND TO .FALSE. OTHERWISE.         HYB 1Ø6Ø
C                                                                      HYB 1Ø7Ø
C        MATV SHOULD BE SET TO .TRUE. IF THE V MATRIX IN THE           HYB 1Ø8Ø
C          DECOMPOSITION IS DESIRED, AND TO .FALSE. OTHERWISE.         HYB 1Ø9Ø
C                                                                      HYB 11ØØ
C        WHEN HYBSVD IS USED TO COMPUTE THE MINIMAL LENGTH LEAST       HYB 111Ø
C        SQUARES SOLUTION TO AN OVERDETERMINED SYSTEM, MATU SHOULD     HYB 112Ø
C        BE SET TO .FALSE. , AND MATV SHOULD BE SET TO .TRUE.          HYB 113Ø
C                                                                      HYB 114Ø
C     ON OUTPUT:                                                       HYB 115Ø
C                                                                      HYB 116Ø
C        A IS UNALTERED (UNLESS OVERWRITTEN BY U OR V).                HYB 117Ø
C                                                                      HYB 118Ø
C        W CONTAINS THE (NON-NEGATIVE) SINGULAR VALUES OF A (THE       HYB 119Ø
C          DIAGONAL ELEMENTS OF W).  THEY ARE SORTED IN DESCENDING     HYB 12ØØ
C          ORDER.  IF AN ERROR EXIT IS MADE, THE SINGULAR VALUES       HYB 121Ø
C          SHOULD BE CORRECT AND SORTED FOR INDICES IERR+1,...,MIN(M,N).HYB 122Ø
C                                                                      HYB 123Ø
C        U CONTAINS THE MATRIX U (ORTHOGONAL COLUMN VECTORS) OF THE    HYB 124Ø
C          DECOMPOSITION IF MATU HAS BEEN SET TO .TRUE.  IF MATU IS    HYB 125Ø
C          FALSE, THEN U IS EITHER USED AS A TEMPORARY STORAGE (IF     HYB 126Ø
C          M .GE. N) OR NOT REFERENCED (IF M .LT. N).                  HYB 127Ø
C          U MAY COINCIDE WITH A IN THE CALLING SEQUENCE.              HYB 128Ø
C          IF AN ERROR EXIT IS MADE, THE COLUMNS OF U CORRESPONDING    HYB 129Ø
C          TO INDICES OF CORRECT SINGULAR VALUES SHOULD BE CORRECT.    HYB 13ØØ
C                                                                      HYB 131Ø
C        V CONTAINS THE MATRIX V (ORTHOGONAL) OF THE DECOMPOSITION IF  HYB 132Ø
C          MATV HAS BEEN SET TO .TRUE.  IF MATV IS                     HYB 133Ø
C          FALSE, THEN V IS EITHER USED AS A TEMPORARY STORAGE (IF     HYB 134Ø
C          M .LT. N) OR NOT REFERENCED (IF M .GE. N).                  HYB 135Ø
C          IF M .GE. N, V MAY ALSO COINCIDE WITH A.  IF AN ERROR       HYB 136Ø
C          EXIT IS MADE, THE COLUMNS OF V CORRESPONDING TO INDICES OF  HYB 137Ø
C          CORRECT SINGULAR VALUES SHOULD BE CORRECT.                  HYB 138Ø
C                                                                      HYB 139Ø
C        Z CONTAINS THE MATRIX X IN THE SINGULAR VALUE DECOMPOSITION   HYB 14ØØ
C                T                                                     HYB 141Ø
C          OF R=XSY,  IF THE MODIFIED ALGORITHM IS USED. IF THE        HYB 142Ø
C          GOLUB-REINSCH PROCEDURE IS USED, THEN IT IS NOT REFERENCED. HYB 143Ø
C          IF MATU HAS BEEN SET TO .FALSE. IN THE CASE M.GE.N (OR      HYB 144Ø
C          MATV SET TO .FALSE. IN THE CASE M.LT.N), THEN Z IS NOT      HYB 145Ø
C          REFERENCED AND NO EXTRA STORAGE IS REQUIRED.                HYB 146Ø
C                                                                      HYB 147Ø
C        IERR IS SET TO                                                HYB 148Ø
C          ZERO       FOR NORMAL RETURN,                              HYB 149Ø
C          K          IF THE K-TH SINGULAR VALUE HAS NOT BEEN         HYB 15ØØ
C                     DETERMINED AFTER 3Ø ITERATIONS.                 HYB 151Ø
C          -1         IF IRHS .LT. Ø .                                HYB 152Ø
C          -2         IF M .LT. 1 .OR. N .LT. 1                       HYB 153Ø
C          -3         IF NA .LT. M .OR. NU .LT. M .OR. NB .LT. M.     HYB 154Ø
C          -4         IF NV .LT. N .                                  HYB 155Ø
C          -5         IF NZ .LT. MIN(M,N).                            HYB 156Ø
C                                                                      HYB 157Ø
C        RV1 IS A TEMPORARY STORAGE ARRAY OF LENGTH AT LEAST MIN(M,N). HYB 158Ø
C                                                                      HYB 159Ø
C     PROGRAMMED BY : TONY CHAN                                        HYB 16ØØ
C                     BOX 2158, YALE STATION,                          HYB 161Ø
C                     COMPUTER SCIENCE DEPT, YALE UNIV.,               HYB 162Ø
C                     NEW HAVEN, CT Ø652Ø.                             HYB 163Ø
C     LAST MODIFIED : JANUARY, 1982.                                   HYB 164Ø
C                                                                      HYB 165Ø
C     HYBSVD USES THE FOLLOWING FUNCTIONS AND SUBROUTINES.             HYB 166Ø
C       INTERNAL  GRSVD, MGNSVD, SRELPR                                HYB 167Ø
C       FORTRAN   MINØ,ABS,SQRT,FLOAT,SIGN,AMAX1                       HYB 168Ø
C       BLAS      SSWAP                                                HYB 169Ø
C                                                                      HYB 17ØØ
C     ---------------------------------------------------------------- HYB 171Ø
```

ALGORITHM 582
The Gibbs-Poole-Stockmeyer and Gibbs-King Algorithms for Reordering Sparse Matrices

JOHN G. LEWIS
Boeing Computer Services Co.

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*sparse and very large systems*; G.m [**Mathematics of Computing**]: Miscellaneous—*FORTRAN*

General Terms: Algorithms

Additional Key Words and Phrases: Matrix bandwidth, matrix profile, matrix wavefront, banded matrix, Gibbs-Poole-Stockmeyer algorithm, Gibbs-King algorithm

DESCRIPTION

Given the structure of a symmetric or structurally symmetric sparse matrix, GPSKCA attempts to find a symmetric reordering of the matrix that produces a smaller bandwidth or profile. References [1], [4], [5], and [6] explain in detail the algorithms realized by GPSKCA. This algorithm provides the same mathematical capabilities as provided by REDUCE, Algorithms 508, and 509, but requires less memory and time and removes some implicit restrictions on the matrices that can be reordered. A description of the differences in the implementation and their effects is given in [7]; GPSKCA and REDUCE produce the same bandwidth and profile on all problems for which REDUCE executes successfully.

The package of subroutines is evoked by the FORTRAN statement

```
CALL GPSKCA (N, NZ, CONNEC, RSTART, DEGREE, OPTPRO, PERMUT, WORK,
             WRKLEN, ERROR, SPACE)
```

where the parameters are described in the listing given here. The subroutines attempt to find a symmetric (row and column) permutation that reduces the bandwidth or profile of the reordered matrix. Whether to emphasize profile reduction or bandwidth reduction is determined by the logical parameter OPTPRO.

A common use for this algorithm will be prior to the solution of sparse linear algebraic equations or algebraic eigenvalue problems of moderately large size. The algorithm is supplied together with a FORTRAN program that accepts a sparse symmetric matrix in a standard sparse format and produces the equivalent reordered matrix in a format suitable for solving linear algebraic equations with the banded solvers in LINPACK [2] or the envelope solver in SPARSPAK [3].

The algorithm has been written in 1966 American National Standard FORTRAN. The package consists of 17 subroutines, GPSKCA, GPSKCB, ...,

Received 28 May 1980; revised 6 January 1982; accepted 10 January 1982
Author's address: Boeing Computer Services Co., Mail Stop 9C-01, 565 Andover Park West, Tukwila, WA 98188.

GPSKCQ. Special comment cards have been included so that the user can easily convert the code to an IBM FORTRAN version that uses half-length integers (INTEGER * 2) for all of the arrays except RSTART. This reduces the memory requirements by nearly a factor of 2 at the cost of restricting the order of the matrix to 32,767 or smaller. There would be no immediate restriction on the number of nonzeros in the matrix.

The primary difference in the usage of GPSKCA and its predecessor, REDUCE, is in the representation of the structure of the matrix. The format used herein is often much more efficient in space than that used in REDUCE. For the convenience of those users of REDUCE who would like to obtain the faster execution speeds of GPSKCA, but who cannot conveniently convert to the new, compact, format, an alternative version, GPSKRA, which accepts the older format for the matrix, will be supplied by the author upon request (not by the ACM distribution service).

REFERENCES

1. CRANE, H.L., JR., GIBBS, N.E., POOLE, W.G., JR., AND STOCKMEYER, P.K. Matrix bandwidth and profile reduction. *ACM Trans. Math. Softw. 2*, 4(Dec. 1976), 375–377.
2. DONGARRA, J.J., BUNCH, J.R., MOLER, C.B., AND STEWART, G.W. *Linpack User's Guide.* Society for Industrial and Applied Mathematics, Philadelphia, 1979.
3. GEORGE, A., LIU, J., AND NG, E. *User Guide for Sparspak: Waterloo Sparse Linear Equations Package.* Dep. Computer Science, Univ. Waterloo, Waterloo, Ont., Canada, 1979.
4. GIBBS, N.E. A hybrid profile reduction algorithm. *ACM Trans. Math. Softw. 2*, 4(Dec. 1976), 378–387.
5. GIBBS, N.E., POOLE, W.G., JR., AND STOCKMEYER, P.K. A comparison of several bandwidth and profile reduction algorithms. *ACM Trans. Math. Softw. 2*, 4(Dec. 1976), 322–330.
6. GIBBS, N.E., POOLE, W.G., JR., AND STOCKMEYER, P.K. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM J. Numer. Anal. 13*, 2(April 1976), 236–250.
7. LEWIS, J.G. Implementation of the Gibbs–Poole–Stockmeyer and Gibbs–King Algorithms. *ACM Trans. Math. Softw. 8*, 2 (June 1982), 180–189.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service.]

```
C === SEPARATOR ===  BEGINNING OF GPS AND GK ALGORITHMS
      SUBROUTINE   GPSKCA   (N, DEGREE, RSTART, CONNEC, OPTPRO, WRKLEN, GPSKCA 1
     1                       PERMUT, WORK, BANDWD, PROFIL, ERROR, SPACE)GPSKCA 2
C                                                                       GPSKCA 3
C     ================================================================GPSKCA 4
C     ================================================================GPSKCA 5
C     =                                                              =GPSKCA 6
C     = B A N D W I D T H    OR    P R O F I L E    R E D U C T I O N  =GPSKCA 7
C     =        FOR A SPARSE AND (STRUCTURALLY) SYMMETRIC MATRIX,      =GPSKCA 8
C     =                          USING EITHER                        =GPSKCA 9
C     =                                                              =GPSKCA1Ø
C     =   THE GIBBS-POOLE-STOCKMEYER ALGORITHM (BANDWIDTH REDUCTION)   =GPSKCA11
C     =                               OR                             =GPSKCA12
C     =          THE GIBBS-KING ALGORITHM (PROFILE REDUCTION)         =GPSKCA13
C     =                                                              =GPSKCA14
C     ================================================================GPSKCA15
C     ================================================================GPSKCA16
C     =     THIS CODE SUPERSEDES TOMS ALGORITHMS 5Ø8 AND 5Ø9 IN THE    =GPSKCA17
C     =     COLLECTED ALGORITHMS OF THE ACM (CALGO).                   =GPSKCA18
C     ================================================================GPSKCA19
C     ================================================================GPSKCA2Ø
C                                                                       GPSKCA21
C     ------------------                                                GPSKCA22
C     P A R A M E T E R S                                               GPSKCA23
C     ------------------                                                GPSKCA24
C                                                                       GPSKCA25
      INTEGER     N, RSTART(N), WRKLEN, BANDWD, PROFIL, ERROR, SPACE    GPSKCA26
C                                                                       GPSKCA27
CIBM  INTEGER *2  DEGREE(N), CONNEC(1), PERMUT(N), WORK(WRKLEN),        GPSKCA28
      INTEGER     DEGREE(N), CONNEC(1), PERMUT(N), WORK(WRKLEN)         GPSKCA29
C                                                                       GPSKCA3Ø
```

```
      LOGICAL     OPTPRO                                                  GPSKCA31
C                                                                         GPSKCA32
C     ------------------------------------------------------------------GPSKCA33
C                                                                         GPSKCA34
C     INPUT PARAMETERS:                                                   GPSKCA35
C     ----- ----------                                                   GPSKCA36
C                                                                         GPSKCA37
C         N       -- THE DIMENSION OF THE MATRIX                        GPSKCA38
C                                                                         GPSKCA39
C         DEGREE,                                                       GPSKCA4Ø
C         RSTART,                                                       GPSKCA41
C         CONNEC -- DESCRIBE THE STRUCTURE OF THE SPARSE MATRIX.        GPSKCA42
C                   DEGREE(I) SPECIFIES THE NUMBER OF NON-ZERO          GPSKCA43
C                   OFF-DIAGONAL ENTRIES IN THE  I-TH  ROW OF THE       GPSKCA44
C                   SPARSE MATRIX.  THE COLUMN INDICES OF THESE         GPSKCA45
C                   ENTRIES ARE GIVEN IN CONSECUTIVE LOCATIONS IN       GPSKCA46
C                   CONNEC, STARTING AT LOCATION  RSTART(I).            GPSKCA47
C                   IN OTHER WORDS, THE INDICES OF THE NON-ZERO         GPSKCA48
C                   OFF-DIAGONAL ELEMENTS OF THE  I-TH  ROW ARE FOUND   GPSKCA49
C                   IN:                                                 GPSKCA5Ø
C                       CONNEC (RSTART(I)),                             GPSKCA51
C                       CONNEC (RSTART(I) + 1),                         GPSKCA52
C                                 . . .                                 GPSKCA53
C                       CONNEC (RSTART(I) + DEGREE(I) - 1)              GPSKCA54
C                                                                         GPSKCA55
C                   DIMENSIONS:                                         GPSKCA56
C                       RSTART IS DIMENSION  N  (OR LONGER).            GPSKCA57
C                       DEGREE IS DIMENSION  N  (OR LONGER).            GPSKCA58
C                       CONNEC IS DIMENSION ROUGHLY THE NUMBER OF NON-  GPSKCA59
C                             ZERO ENTRIES IN THE MATRIX.               GPSKCA6Ø
C                                                                         GPSKCA61
C         OPTPRO -- .TRUE. IF REDUCING THE PROFILE OF THE MATRIX        GPSKCA62
C                      IS MORE IMPORTANT THAN REDUCING THE              GPSKCA63
C                      BANDWIDTH                                        GPSKCA64
C                   .FALSE. IF BANDWIDTH REDUCTION IS MOST IMPORTANT    GPSKCA65
C                                                                         GPSKCA66
C         WRKLEN -- THE  ACTUAL  LENGTH OF THE VECTOR  WORK  AS SUPPLIEDGPSKCA67
C                   BY THE USER.  SEE THE DISCUSSION OF THE WORKSPACE   GPSKCA68
C                   'WORK'  BELOW FOR TYPICAL STORAGE REQUIREMENTS.     GPSKCA69
C                   THE VALUE OF  WRKLEN  WILL BE USED TO ENSURE THAT   GPSKCA7Ø
C                   THE ROUTINE WILL NOT USE MORE STORAGE THAN IS       GPSKCA71
C                   AVAILABLE.  IF NOT ENOUGH SPACE IS GIVEN IN  WORK   GPSKCA72
C                   TO PERMIT A SOLUTION TO BE FOUND, THE  ERROR  FLAG  GPSKCA73
C                   WILL BE SET AND FURTHER COMPUTATION STOPPED.        GPSKCA74
C                                                                         GPSKCA75
C                                                                         GPSKCA76
C     INPUT AND OUTPUT PARAMETER:                                         GPSKCA77
C     ----- --- ------ ---------                                         GPSKCA78
C                                                                         GPSKCA79
C         PERMUT -- ON INPUT, AN ALTERNATIVE REORDERING FOR THE         GPSKCA8Ø
C                   ROWS AND COLUMNS OF THE MATRIX.  PERMUT(I) GIVES    GPSKCA81
C                   THE POSITION IN WHICH ROW AND COLUMN  I  SHOULD     GPSKCA82
C                   BE PLACED TO REDUCE THE BANDWIDTH OR THE PROFILE.   GPSKCA83
C                   IF THE USER HAS NO ALTERNATIVE TO THE NATURAL       GPSKCA84
C                   ORDERING IMPLICIT IN  DEGREE,  RSTART  AND  CONNEC, GPSKCA85
C                   HE SHOULD INITIALIZE  PERMUT  TO BE THE IDENTITY    GPSKCA86
C                   PERMUTATION  PERMUT(I) = I .                        GPSKCA87
C                                                                         GPSKCA88
C                   ON OUTPUT,  PERMUT  WILL CONTAIN THE PERMUTATION    GPSKCA89
C                   FOR REORDERING THE ROWS AND COLUMNS WHICH REDUCES   GPSKCA9Ø
C                   THE BANDWIDTH AND/OR PROFILE.  THE RESULT WILL BE   GPSKCA91
C                   THE REORDERING FOUND BY 'GPSKCA' OR THE REORDERING  GPSKCA92
C                   GIVEN BY THE USER IN 'PERMUT', WHICHEVER DOES THE   GPSKCA93
C                   JOB BETTER.                                         GPSKCA94
C                                                                         GPSKCA95
C                                                                         GPSKCA96
C     OUTPUT PARAMETERS:                                                  GPSKCA97
C     ------ ----------                                                  GPSKCA98
C                                                                         GPSKCA99
C         WORK   -- A TEMPORARY STORAGE VECTOR, OF LENGTH SOMEWHAT      GPSKC1ØØ
C                   GREATER THAN  3N.  THE SPACE BEYOND  3N  REQUIRED   GPSKC1Ø1
C                   IS PROBLEM-DEPENDENT.  ANY PROBLEM CAN BE SOLVED    GPSKC1Ø2
C                   IN  6N+3  LOCATIONS.                                GPSKC1Ø3
```

```
C                MOST PROBLEMS CAN BE REORDERED WITH  4N  LOCATIONS  GPSKC1Ø4
C                IN 'WORK'.  IF SPACE IS NOT A CONSTRAINT, PROVIDE   GPSKC1Ø5
C                6N+3  LOCATIONS IN 'WORK'.  OTHERWISE, PROVIDE AS   GPSKC1Ø6
C                MUCH MORE THAN  3N  AS IS CONVENIENT AND CHECK THE  GPSKC1Ø7
C                ERROR FLAG AND SPACE REQUIRED PARAMETERS (SEE BELOW)GPSKC1Ø8
C                                                                    GPSKC1Ø9
C                ON OUTPUT, THE 1ST  N  LOCATIONS OF WORK WILL BE    GPSKC11Ø
C                A LISTING OF THE ORIGINAL ROW AND COLUMN INDICES AS GPSKC111
C                THEY APPEAR IN THE COMPUTED REORDERING.             GPSKC112
C                LOCATIONS  N+1, ... , 2N  OF  WORK  WILL CONTAIN    GPSKC113
C                THE NEW POSITIONS FOR THE EQUATIONS IN THE ORDER    GPSKC114
C                FOUND BY GPSKCA.  THUS, THE TWO VECTORS ARE INVERSE GPSKC115
C                PERMUTATIONS OF EACH OTHER.  IF THE ORDERING        GPSKC116
C                FOUND BY THIS ALGORITHM IS BETTER THAN THE USER-    GPSKC117
C                SUPPLIED ORDER, THE SECOND PERMUTATION VECTOR IS    GPSKC118
C                IDENTICAL TO THE RESULT RETURNED IN  'PERMUT'.      GPSKC119
C                                                                    GPSKC12Ø
C        BANDWD -- THE BANDWIDTH OF THE MATRIX WHEN ROWS AND COLUMNS GPSKC121
C                ARE REORDERED IN THE ORDERING RETURNED IN  PERMUT.  GPSKC122
C                                                                    GPSKC123
C        PROFIL -- THE PROFILE OF THE MATRIX WHEN ROWS AND COLUMNS ARE GPSKC124
C                REORDERED IN THE ORDERING RETURNED IN  PERMUT.      GPSKC125
C                                                                    GPSKC126
C        ERROR  -- WILL BE EQUAL TO ZERO IF A NEW NUMBERING COULD BE GPSKC127
C                FOUND IN THE SPACE PROVIDED.  OTHERWISE,  ERROR     GPSKC128
C                WILL BE SET TO A POSITIVE ERROR CODE (SEE TABLE     GPSKC129
C                GIVEN BELOW).  IF THE REORDERING ALGORITHM HAS BEEN GPSKC13Ø
C                STOPPED BY LACK OF WORKSPACE, THE SPACE PARAMETER   GPSKC131
C                WILL BE SET TO THE NUMBER OF ADDITIONAL LOCATIONS   GPSKC132
C                REQUIRED TO COMPLETE AT LEAST THE NEXT PHASE OF     GPSKC133
C                THE ALGORITHM.                                      GPSKC134
C                                                                    GPSKC135
C                WHENEVER A NON-ZERO VALUE FOR  ERROR  IS GIVEN      GPSKC136
C                PERMUT  WILL RETAIN THE VALUES PROVIDED BY THE USER GPSKC137
C                AND THE SCALARS  BANDWD  AND  PROFIL  WILL BE SET TOGPSKC138
C                OUTRAGEOUS VALUES.  IT IS THE USER'S RESPONSIBILITY GPSKC139
C                TO CHECK THE STATUS OF  ERROR.                      GPSKC14Ø
C                                                                    GPSKC141
C        SPACE  -- WILL INDICATE EITHER HOW MUCH SPACE THE REORDERING GPSKC142
C                ACTUALLY REQUIRED OR HOW MUCH SPACE WILL BE         GPSKC143
C                REQUIRED TO COMPLETE THE NEXT PHASE OF THE          GPSKC144
C                REORDERING ALGORITHM.  THE POSSIBLE OUTCOMES ARE .. GPSKC145
C                                                                    GPSKC146
C                   ERROR = Ø          SPACE IS THE MINIMAL VALUE FORGPSKC147
C                                      WRKLEN  REQUIRED TO REORDER   GPSKC148
C                                      THIS MATRIX AGAIN.            GPSKC149
C                                                                    GPSKC15Ø
C                   ERROR FJ Ø         SPACE IS THE MINIMUM NUMBER   GPSKC151
C                   DUE TO LACK OF     OF EXTRA WORKSPACE REQUIRED   GPSKC152
C                   WORKSPACE          TO CONTINUE THE REORDERING    GPSKC153
C                                      ALGORITHM ON THIS MATRIX.     GPSKC154
C                                                                    GPSKC155
C                   ERROR FJ Ø         SPACE = -1                    GPSKC156
C                   DUE TO ERROR                                     GPSKC157
C                   IN DATA STRUCTURES                               GPSKC158
C                                                                    GPSKC159
C                                                                    GPSKC16Ø
C    ================================================================GPSKC161
C                                                                    GPSKC162
C    ---------------------                                           GPSKC163
C    E R R O R    C O D E S                                          GPSKC164
C    ---------------------                                           GPSKC165
C                                                                    GPSKC166
C        ERROR CODES HAVE THE FORM  ØXY  OR  1XY.                    GPSKC167
C                                                                    GPSKC168
C        ERRORS OF THE FORM  1XY  RESULT FROM INADEQUATE WORKSPACE.  GPSKC169
C                                                                    GPSKC17Ø
C        ERRORS OF THE FORM  ØXY  ARE INTERNAL PROGRAM CHECKS, WHICH GPSKC171
C        MOST LIKELY OCCUR BECAUSE THE CONNECTIVITY STRUCTURE OF THE GPSKC172
C        MATRIX IS REPRESENTED INCORRECTLY (E.G., THE DEGREE OF      GPSKC173
C        A NODE IS NOT CORRECT  OR  NODE I IS CONNECTED TO NODE J,   GPSKC174
C        BUT NOT CONVERSELY).                                        GPSKC175
```

583-P 1- 0

ALGORITHM 583
LSQR: Sparse Linear Equations and Least Squares Problems

CHRISTOPHER C. PAIGE
McGill University, Canada
and
MICHAEL A. SAUNDERS
Stanford University

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*linear systems (direct and iterative methods)*; G.3 [**Mathematics of Computing**]: Probability and Statistics—*statistical computing; statistical software*; G.m [**Mathematics of Computing**]: Miscellaneous—*FORTRAN program units*

General Terms: Algorithms

Additional Key Words and Phrases: Analysis of variance, conjugate-gradient method, least squares, linear equations, regression, sparse matrix

1. INTRODUCTION

LSQR finds a solution x to the following problems:

Unsymmetric equations:	solve	$Ax = b$	(1.1)
Linear least squares:	minimize	$\|Ax - b\|_2$	(1.2)
Damped least squares:	minimize	$\left\| \begin{bmatrix} A \\ \lambda I \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2$	(1.3)

where A is a matrix with m rows and n columns, b is an m-vector, λ is a scalar, and the given data A, b, λ are real. The matrix A will normally be large and sparse. It is defined by means of a user-written subroutine APROD, whose essential function is to compute products of the form Ax and $A^T y$ for given vectors x and y.

Problems (1.1) and (1.2) are treated as special cases of (1.3), which we shall write as

$$\min \|\bar{A}\, x - \bar{b}\|_2, \qquad \bar{A} = \begin{bmatrix} A \\ \lambda I \end{bmatrix}, \qquad \bar{b} = \begin{bmatrix} b \\ 0 \end{bmatrix}. \tag{1.4}$$

Received 4 June 1980; revised 23 September 1981; accepted 28 February 1982
This work was supported by Natural Sciences and Engineering Research Council of Canada Grant A8652; by the New Zealand Department of Scientific and Industrial Research; and by U.S. National Science Foundation Grants MCS-7926009 and ECS-8012974, the Department of Energy under Contract AM03-76SF00326, PA No. DE-AT03-76ER72018, the Office of Naval Research under Contract N00014-75-C-0267, and the Army Research Office under Contract DAA29-79-C-0110.
Authors' addresses: C. C. Paige, School of Computer Science, McGill University, Montreal, Quebec, Canada H3A 2K6; M. A. Saunders, Department of Operations Research, Stanford University, Stanford, CA 94305.

Table I. Comparison of CGLS and LSQR

	Storage	Work per iteration
CGLS, $\lambda = 0$	$2m + 2n$	$2m + 3n$
CGLS, $\lambda \neq 0$	$2m + 2n$	$2m + 5n$
LSQR, any λ	$m + 2n$	$3m + 5n$

An earlier successful method for such problems is the *conjugate-gradient method for least squares systems* given by Hestenes and Stiefel [3]. (This method is described as algorithm CGLS in [6, sect. 7.1].) CGLS and LSQR are iterative methods with similar qualitative properties. Their computational requirements are summarized in Table I. In addition they require a product Ax and a product A^Ty each iteration.

In order to achieve the storage shown for LSQR, we ask the user to implement the matrix-vector products in the form

$$y \leftarrow y + Ax \qquad \text{and} \qquad x \leftarrow x + A^Ty, \tag{1.5}$$

where $\leftarrow$ means that one of the given vectors is overwritten by the expression shown. (A parameter specifies which expression the user's subroutine APROD should compute on any given entry.) We see that LSQR has a storage advantage if the operations (1.5) can be performed with no additional storage beyond that required to represent A. For least squares applications with many observations ($m \gg n$), this could be useful.

The work shown in Table I is the number of floating-point multiplications per iteration, excluding the work involved in the products Ax, A^Ty. Since CGLS is somewhat more efficient, we would not discourage using that method whenever A or $\bar{A}$ is well conditioned. However, LSQR is likely to obtain a more accurate solution in fewer iterations if $\bar{A}$ is moderately or severely ill-conditioned.

Let $\bar{r}_k = \bar{b} - \bar{A}x_k$ be the residual vector associated with the kth iteration. LSQR provides estimates of $\| x_k \|_2$, $\| \bar{r}_k \|_2$, $\| \bar{A}^T\bar{r}_k \|_2$, the norm of $\bar{A}$, the condition number of $\bar{A}$, and standard errors for the components of x. The last two items require a further $2n$ multiplications per iteration and an additional n-vector of storage.

Subroutine LSQR is written in the PFORT subset of American National Standard FORTRAN. It contains no machine-dependent constants. Auxiliary routines required are APROD, NORMLZ, SCOPY, SNRM2, and SSCAL. The last three correspond to members of the BLAS collection [5].

2. MATHEMATICAL BACKGROUND

Algorithmic details are given in [6], mainly for the case $\lambda = 0$. We summarize these here with λ reintroduced, and show that a given value of λ may be dealt with at negligible cost. The vector norm $\| v \|_2 = (v^Tv)^{1/2}$ is used throughout.

LSQR uses an algorithm of Golub and Kahan to reduce A to lower bidiagonal form. The quantities produced from A and b after $k + 1$ steps of the bidiagonalization (procedure Bidiag 1 [6]) are

$$\begin{aligned} \beta_1 &= \| b \|, \\ U_{k+1} &= [u_1, u_2, \ldots, u_{k+1}], \\ V_{k+1} &= [v_1, v_2, \ldots, v_{k+1}], \end{aligned} \qquad B_k = \begin{bmatrix} \alpha_1 & & & & \\ \beta_2 & \alpha_2 & & & \\ & \beta_3 & \ddots & & \\ & & \ddots & \alpha_k & \\ & & & \beta_{k+1} \end{bmatrix}. \tag{2.1}$$

The kth approximation to the solution x is then defined to be $x_k = V_k y_k$, where y_k solves the subproblem

$$\min \left\| \begin{bmatrix} B_k \\ \lambda I \end{bmatrix} y_k - \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix} \right\|. \tag{2.2}$$

Letting the associated residual vectors be

$$\begin{aligned} t_{k+1} &= \beta_1 e_1 - B_k y_k \\ r_k &= b - A x_k \\ \bar{r}_k &= \bar{b} - \bar{A} x_k, \end{aligned} \tag{2.3}$$

we find that the relations

$$\begin{aligned} r_k &= U_{k+1} t_{k+1} \\ A^{\mathrm{T}} r_k &= \lambda^2 x_k + \alpha_{k+1} \tau_{k+1} v_{k+1} \end{aligned} \tag{2.4}$$

will hold to machine accuracy, where τ_{k+1} is the last component of t_{k+1}, and we therefore conclude that (r_k, x_k) will be an acceptable solution of (1.4) if the computed value of either $\| t_{k+1} \|$ or $| \alpha_{k+1} \tau_{k+1} |$ is suitably small.

Björck [1] has previously observed that subproblem (2.2) is the appropriate generalization of $\min \| B_k y_k - \beta_1 e_1 \|$, when $\lambda \neq 0$. He also discusses methods for computing y_k and x_k efficiently for various λ and k.

In LSQR we assume that a single value of λ is given, and to save storage and work, we do not compute y_k, r_k, or t_{k+1}. The orthogonal factorization

$$Q_k \begin{bmatrix} B_k & \beta_1 e_1 \\ \lambda I & 0 \end{bmatrix} = \begin{bmatrix} R_k & f_k \\ 0 & \bar{\phi}_{k+1} \\ 0 & q_k \end{bmatrix} \tag{2.5}$$

is computed ($Q_k^{\mathrm{T}} Q_k = I$; R_k upper bidiagonal, $k \times k$) and this would give $R_k y_k = f_k$, but instead we solve $R_k^{\mathrm{T}} D_k^{\mathrm{T}} = V_k^{\mathrm{T}}$ and form $x_k = D_k f_k$.

The factorization (2.5) is formed similarly to the case $\lambda = 0$ in [6], except that two rotations are required per step instead of one. For $k = 2$, the factorization proceeds according to

$$\begin{bmatrix} \alpha_1 & & \beta_1 \\ \beta_2 & \alpha_2 & \\ & \beta_3 & \\ \lambda & & \\ & \lambda & \end{bmatrix} \rightarrow \begin{bmatrix} \tilde{\rho}_1 & & \tilde{\phi}_1 \\ \beta_2 & \alpha_2 & \\ & \beta_3 & \\ & & \psi_1 \\ & \lambda & \end{bmatrix} \rightarrow \begin{bmatrix} \rho_1 & \theta_2 & \phi_1 \\ & \bar{\rho}_2 & \bar{\phi}_2 \\ & \beta_3 & \\ & & \psi_1 \\ & \lambda & \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} \rho_1 & \theta_2 & \phi_1 \\ & \tilde{\rho}_2 & \tilde{\phi}_2 \\ & \beta_3 & \\ & & \psi_1 \\ & & \psi_2 \end{bmatrix} \rightarrow \begin{bmatrix} \rho_1 & \theta_2 & \phi_1 \\ & \rho_2 & \phi_2 \\ & & \bar{\phi}_3 \\ & & \psi_1 \\ & & \psi_2 \end{bmatrix}.$$

Note that the first λ is rotated into the diagonal element α_1. This alters the right-hand side $\beta_1 e_1$ to produce ψ_1, the first component of q_k. An alternative is to rotate λ into β_2 (and similarly for later λ), since this does not affect the right-hand side and it more closely simulates the algorithm that results when LSQR is applied to $\bar{A}$ and $\bar{b}$ directly. However, the rotations then have a greater effect on B_k, and in practice the first option has proved to give marginally more accurate results.

The estimates required to implement the stopping criteria are

$$\| \bar{r}_k \|^2 = \| r_k \|^2 + \lambda^2 \| x_k \|^2 \approx \bar{\phi}_{k+1}^2 + \| q_k \|^2,$$

$$\| \bar{A}^{\mathrm{T}} \bar{r}_k \| = \| A^{\mathrm{T}} r_k - \lambda^2 x_k \| \approx \left| \frac{\alpha_{k+1} \beta_{k+1} \phi_k}{\rho_k} \right|.$$

This is a simple generalization of the case $\lambda = 0$. No additional storage is needed for q_k, since only its norm is required. In short, although the presence of λ complicates the algorithm description, it adds essentially nothing to the storage and work per iteration.

3. REGULARIZATION AND RELATED WORK

Introducing λ as in (1.3) is just one way of "regularizing" the solution x, in the sense that it can reduce the size of the computed solution and make its components less sensitive to changes in the data. LSQR is applicable when a value of λ is known a priori. The value is entered via the subroutine parameter DAMP. A second method for regularizing x is available through LSQR's parameter ACOND, which can cause iterations to terminate before $\|x_k\|$ becomes large. A similar approach has recently been described by Wold et al. [9], who give an illuminating interpretation of the bidiagonalization as a partial least squares procedure. Their description will also be useful to those who prefer the notation of multiple regression.

Methods for choosing λ, and other approaches to regularization, are given in [1, 2, 4, 8] and elsewhere. For a philosophical discussion, see [7].

4. CODING APROD

The best way to compute $y + Ax$ and $x + A^Ty$ depends upon the origin of the matrix A. We shall illustrate a case that commonly arises, in which A is a sparse matrix whose nonzero coefficients are stored by *rows* in a simple list. Let A have M rows, N columns, and NZ nonzeros. Conceptually we need three arrays dimensioned as REAL RA(NZ) and INTEGER JA(NZ), NA(M), where

RA(L) is the Lth nonzero of A, counting across row 1, then across row 2, and so on;
JA(L) is the column in which the Lth nonzero of A lies;
NA(I) is the number of nonzero coefficients in the Ith row of A.

These quantities may be used in a straightforward way, as shown in Figure 1 (a FORTRAN implementation). We assume that they are made available to APROD through COMMON, and that the actual array dimensions are suitably large.

Blank or labeled COMMON will often be convenient for transmitting data to APROD. (Of course, some of the data could be local to APROD.) For greater generality, the parameter lists for LSQR and APROD include two workspace arrays IW, RW and their lengths LENIW, LENRW. LSQR does not use these parameters directly; it just passes them to APROD.

Figure 2 illustrates their use on the same example (sparse A stored by rows). An auxiliary subroutine APROD1 is needed to make the code readable. A similar scheme should be used to initialize the workspace parameters prior to calling LSQR.

Returning to the example itself, it may often be natural to store A by *columns* rather than rows, using analogous data structures. However, we note that in sparse least squares applications, A may have many more rows than columns ($M \gg N$). In such cases it is vital to store A by rows as shown, if the machine being used has a paged (virtual) memory. Random access is then restricted to arrays of length N rather than M, and page faults will therefore be kept to a minimum.

Note also that the arrays RA, JA, NA are adequate for computing both Ax and A^Ty; we do not need to store A by rows *and* by columns.

Regardless of the application, it will be apparent when coding APROD for the two values of MODE that the matrix A is effectively being defined *twice*. Great care must be taken to avoid coding inconsistent expressions $y + A_1x$ and $x + A_2^Ty$, where either A_1 or A_2 is different from the desired A. (If $A_1 \neq A_2$, algorithm LSQR will not converge.) Parameters ANORM, ACOND, and CONLIM provide a safeguard for such an event.

5. PRECONDITIONING

It is well known that conjugate-gradient methods can be accelerated if a nonsingular matrix M is available to approximate A in some useful sense. When A is

square and nonsingular, the system $Ax = b$ is equivalent to both of the following systems:

$$(M^{-1}A)x = c \quad \text{where} \quad Mc = b; \tag{5.1}$$

$$(AM^{-1})z = b \quad \text{where} \quad Mx = z. \tag{5.2}$$

For least squares systems (undamped), only the analogue of (5.2) is applicable:

$$\min\|Ax - b\|_2 = \min\|(AM^{-1})z - b\|_2, \quad \text{where} \quad Mx = z. \tag{5.3}$$

```
      SUBROUTINE APROD( MODE,M,N,X,Y,
     *                  LENIW,LENRW,IW,RW )
C
      INTEGER    MODE,M,N,LENIW,LENRW
      INTEGER    IW(LENIW)
      REAL       X(N),Y(M),RW(LENRW)
C
C     APROD  PERFORMS THE FOLLOWING FUNCTIONS:
C
C       IF MODE = 1, SET  Y = Y + A*X
C       IF MODE = 2, SET  X = X + A(TRANSPOSE)*Y
C
C     WHERE  A  IS A MATRIX STORED BY ROWS IN
C     THE ARRAYS  RA, JA, NA.  IN THIS EXAMPLE,
C     RA, JA, NA  ARE STORED IN COMMON.
C
      REAL       RA
      INTEGER    JA,NA
      COMMON     RA(9000),JA(9000),NA(1000)
C
C     INTEGER    I,J,L,L1,L2
      REAL       SUM,YI,ZERO
C
      ZERO = 0.0
      L2   = 0
      IF (MODE .NE.1) GO TO 400
C
C     ----------------------------------------
C     MODE = 1  --  SET  Y = Y + A*X.
C     ----------------------------------------
      DO 200 I = 1, M
         SUM = ZERO
         L1  = L2 + 1
         L2  = L2 + NA(I)
         DO 100 L = L1, L2
            J   = JA(L)
            SUM = SUM + RA(L)*X(J)
  100    CONTINUE
         Y(I) = Y(I) + SUM
  200 CONTINUE
      RETURN
C
C     ----------------------------------------
C     MODE = 2  --  SET  X = X + A(TRANSPOSE)*Y.
C     ----------------------------------------
  400 DO 600 I = 1, M
         YI = Y(I)
         L1 = L2 + 1
         L2 = L2 + NA(I)
         DO 500 L = L1, L2
            J    = JA(L)
            X(J) = X(J) + RA(L)*YI
  500    CONTINUE
  600 CONTINUE
      RETURN
C
C     END OF APROD
      END
```

Fig. 1. Computation of $y + Ax$, $x + A^Ty$, where A is a sparse matrix stored compactly by rows. For convenience, the data structure for A is held in COMMON.

```
      SUBROUTINE APROD( MODE,M,N,X,Y,
     *                  LENIW,LENRW,IW,RW )
C
      INTEGER    MODE,M,N,LENIW,LENRW
      INTEGER    IW(LENIW)
      REAL       X(N),Y(M),RW(LENRW)
C
C     APROD  PERFORMS THE FOLLOWING FUNCTIONS:
C
C       IF MODE = 1, SET  Y = Y + A*X
C       IF MODE = 2, SET  X = X + A(TRANSPOSE)*Y
C
C     WHERE  A  IS A MATRIX STORED BY ROWS IN
C     THE ARRAYS  RA, JA, NA.  IN THIS EXAMPLE,
C     APROD  IS AN INTERFACE BETWEEN  LSQR  AND
C     ANOTHER USER ROUTINE THAT DOES THE WORK.
C     THE WORKSPACE ARRAY RW CONTAINS RA.
C     THE FIRST M COMPONENTS OF IW CONTAIN NA,
C     AND THE REMAINDER OF IW CONTAINS JA.
C     THE DIMENSIONS OF RW AND IW ARE ASSUMED
C     TO BE SUFFICIENTLY LARGE.
C
      INTEGER    LENJA,LENRA,LOCJA
C
      LOCJA  = M + 1
      LENJA  = LENIW - LOCJA + 1
      LENRA  = LENRW
      CALL APROD1( MODE,M,N,X,Y,
     *             LENJA,LENRA,IW,IW(LOCJA),RW )
      RETURN
C
C     END OF APROD
      END



      SUBROUTINE APROD1( MODE,M,N,X,Y,
     *                   LENJA,LENRA,NA,JA,RA )
C
      INTEGER    MODE,M,N,LENJA,LENRA
      INTEGER    NA(M),JA(LENJA)
      REAL       X(N),Y(M),RA(LENRA)
C
C     APROD1  DOES THE WORK FOR  APROD.
C
C     INTEGER    I,J,L,L1,L2
      REAL       SUM,YI,ZERO
C
C
      < the same code as in APROD in Figure 1 >
C
C
C     END OF APROD1
      END
```

Fig. 2. Same as Figure 1, with the data structure for A held in the workspace parameters.

```
LEAST-SQUARES TEST PROBLEM      P(   20    10     1     1     1.00E-03 )

CONDITION NO. =  9.9995E 00     RESIDUAL FUNCTION =  9.812157000E-01


                    LSQR   --   LEAST-SQUARES SOLUTION OF  A*X = B

                    THE MATRIX  A  HAS    20 ROWS   AND    10 COLS
                    THE DAMPING PARAMETER IS    DAMP   =  1.00E-03

                    ATOL   =  1.00E-06          CONLIM =  1.00E 02
                    BTOL   =  1.00E-06          ITNLIM =        80
```

Fig. 3. Example output from test program and LSQR on a damped least squares problem.

```
   ITN        X(1)              FUNCTION       COMPATIBLE INCOMPATIBLE NORM(ABAR) COND(ABAR)

     0   0.0000000000E-01   6.3410580000E 00   1.000E 00   9.135E-02

     1  -6.3564250000E-01   4.0387670000E 00   6.369E-01   7.244E-01   7.51E-01   1.00E 00
     2  -4.7282630000E-01   2.3303970000E 00   3.675E-01   3.349E-01   1.10E 00   2.42E 00
     3  -2.8075080000E-01   1.8962160000E 00   2.990E-01   2.462E-01   1.30E 00   3.63E 00
     4   2.6825070000E-01   1.5905200000E 00   2.508E-01   1.424E-01   1.49E 00   5.27E 00
     5   1.2649560000E 00   1.4032960000E 00   2.213E-01   1.160E-01   1.59E 00   7.09E 00
     6   2.0648040000E 00   1.2910880000E 00   2.036E-01   9.273E-02   1.70E 00   9.03E 00
     7   3.0031450000E 00   1.2072930000E 00   1.904E-01   8.294E-02   1.79E 00   1.12E 01
     8   3.7526340000E 00   1.1551220000E 00   1.822E-01   5.138E-02   1.90E 00   1.34E 01
     9   5.4443550000E 00   1.0780640000E 00   1.700E-01   3.155E-02   1.95E 00   1.72E 01
    10   8.9918140000E 00   9.8151740000E-01   1.548E-01   1.283E-02   1.96E 00   2.44E 01

    11   8.9998990000E 00   9.8120520000E-01   1.547E-01   2.017E-04   2.20E 00   2.75E 01
    12   8.9999290000E 00   9.8120540000E-01   1.547E-01   4.361E-06   2.38E 00   2.98E 01
    13   8.9999280000E 00   9.8120550000E-01   1.547E-01   9.078E-07   2.48E 00   3.13E 01
NO. OF ITERATIONS =    13          STOPPING CONDITION =  2

THE LEAST-SQRS SOLN IS GOOD ENOUGH, GIVEN  ATOL

                    RESIDUAL NORM (ABAR*X - BBAR)   RESIDUAL NORM (NORMAL EQNS)   SOLUTION NORM (X)
                    -----------------------------   ---------------------------   -----------------

ESTIMATED BY LSQR          9.812055E-01                  2.206693E-06                1.688187E 01
COMPUTED FROM X            9.812157E-01                  1.083419E-05                1.688184E 01


SOLUTION
    1   8.99993        2   7.99997        3   6.99998        4   5.99999        5   4.99999
    6   3.99999        7   3.00000        8   2.00000        9   0.999994      10 -0.204206E-05

STANDARD ERRORS
    1   2.11589        2  0.888101        3  0.685644        4  0.556184        5  0.614104
    6  0.409182        7  0.565480        8  0.519385        9  0.375466       10  0.589787
```

Figure 3 (continued)

We note only that subroutine *LSQR may be applied without change* to systems (5.1)-(5.3). The effect of M is localized to the user's own subroutine APROD. For example, when MODE = 1, APROD for the last two systems should compute $y + (AM^{-1})x$ by first solving $Mw = x$ and then computing $y + Aw$. Clearly it must be possible to solve systems involving M and M^T very efficiently.

6. OUTPUT

Subroutine LSQR produces printed output on file NOUT, if the parameter NOUT is positive. This is illustrated in Figure 3, in which the least squares problem solved is $P(20, 10, 1, 1)$ as defined in [6], with a slight generalization to include a damping parameter $\lambda = 10^{-3}$. (Single precision was used on an IBM 370/168.) The items printed at the kth iteration are as follows.

ITN — The iteration number k. Results are always printed for the first 10 and last 10 iterations. Intermediate results are printed if $m \leq 40$ or $n \leq 40$, or if one of the convergence conditions is nearly satisfied. Otherwise, information is printed every 10th iteration.

X(1) — The value of the first element of the approximate solution x_k.

FUNCTION — The value of the function being minimized, namely $\| \bar{r}_k \| = (\| r_k \|^2 + \lambda^2 \| x_k \|^2)^{1/2}$.

COMPATIBLE — A dimensionless quantity which should converge to zero *if and only if* $Ax = b$ is compatible. It is an estimate of $\| \bar{r}_k \| / \| b \|$, which decreases monotonically.

INCOMPATIBLE	A dimensionless quantity which should converge to zero *if and only if* the optimum $\|\bar{r}_k\|$ is nonzero. It is an estimate of $\|\bar{A}^T\bar{r}_k\|/(\|\bar{A}\|_F\|\bar{r}_k\|)$, which is usually *not* monotonic.
NORM(ABAR)	A monotonically increasing estimate of $\|\bar{A}\|_F$.
COND(ABAR)	A monotonically increasing estimate of cond($\bar{A}$) = $\|\bar{A}\|_F\|\bar{A}^+\|_F$, the condition number of $\bar{A}$.

ACKNOWLEDGMENT

The authors are grateful to Richard Hanson for suggestions that prompted several improvements to the implementation of LSQR.

REFERENCES

1. BJÖRCK, Å. A bidiagonalization algorithm for solving ill-posed systems of linear equations. Rep. LITH-MAT-R-80-33, Dep. Mathematics, Linköping Univ., Linköping, Sweden, 1980.
2. ELDÉN, L. Algorithms for the regularization of ill-conditioned least squares problems. *BIT 17* (1977), 134–145.
3. HESTENES, M.R., AND STIEFEL, E. Methods of conjugate gradients for solving linear systems. *J. Res. N.B.S. 49* (1952), 409–436.
4. LAWSON, C.L., AND HANSON, R.J. *Solving Least Squares Problems.* Prentice-Hall, Englewood Cliffs, N.J., 1974.
5. LAWSON, C.L., HANSON, R.J., KINCAID, D.R., AND KROGH, F.T. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Softw. 5*, 3 (Sept. 1979), 308–323 and (Algorithm) 324–325.
6. PAIGE, C.C., AND SAUNDERS, M.A. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Softw. 8*, 1 (March 1982), 43–71.
7. SMITH, G., AND CAMPBELL, F. A critique of some ridge regression methods. *J. Am. Stat. Assoc. 75*, 369 (March 1980), 74–81.
8. VARAH, J.M. A practical examination of some numerical methods for linear discrete ill-posed problems. *SIAM Rev. 21* (1979), 100–111.
9. WOLD, S., WOLD, H., DUNN, W.J., AND RUHE, A. The collinearity problem in linear and nonlinear regression. The partial least squares (PLS) approach to generalized inverses. Rep. UMINF-83.80, Univ. Umeå, Umeå, Sweden, 1980.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service.

```
      SUBROUTINE LSQR( M,N,APROD,DAMP,                                  1.
     1                 LENIW,LENRW,IW,RW,                               2.
     2                 U,V,W,X,SE,                                      3.
     3                 ATOL,BTOL,CONLIM,ITNLIM,NOUT,                    4.
     4                 ISTOP,ANORM,ACOND,RNORM,ARNORM,XNORM )           5.
C                                                                       6.
      EXTERNAL    APROD                                                 7.
      INTEGER     M,N,LENIW,LENRW,ITNLIM,NOUT,ISTOP                     8.
      INTEGER     IW(LENIW)                                             9.
      REAL        RW(LENRW),U(M),V(N),W(N),X(N),SE(N),                  1Ø.
     1            ATOL,BTOL,CONLIM,DAMP,ANORM,ACOND,RNORM,ARNORM,XNORM  11.
C     ------------------------------------------------------------------ 12.
C                                                                       13.
C     LSQR  FINDS A SOLUTION  X  TO THE FOLLOWING PROBLEMS...           14.
C                                                                       15.
C     1. UNSYMMETRIC EQUATIONS --    SOLVE  A*X = B                     16.
C                                                                       17.
C     2. LINEAR LEAST SQUARES  --    SOLVE  A*X = B                     18.
C                                    IN THE LEAST-SQUARES SENSE         19.
C                                                                       2Ø.
C     3. DAMPED LEAST SQUARES  --    SOLVE  (   A    )*X = ( B )        21.
C                                           ( DAMP*I )     ( Ø )        22.
C                                    IN THE LEAST-SQUARES SENSE         23.
C                                                                       24.
C     WHERE  A  IS A MATRIX WITH  M  ROWS AND  N  COLUMNS, B  IS AN     25.
C     M-VECTOR, AND  DAMP  IS A SCALAR (ALL QUANTITIES REAL).           26.
C     THE MATRIX  A  IS INTENDED TO BE LARGE AND SPARSE.  IT IS ACCESSED 27.
C     BY MEANS OF SUBROUTINE CALLS OF THE FORM                          28.
```

```
C                                                                       29.
C              CALL APROD( MODE,M,N,X,Y,LENIW,LENRW,IW,RW )             3Ø.
C                                                                       31.
C     WHICH MUST PERFORM THE FOLLOWING FUNCTIONS...                     32.
C                                                                       33.
C              IF MODE = 1, COMPUTE  Y = Y + A*X.                       34.
C              IF MODE = 2, COMPUTE  X = X + A(TRANSPOSE)*Y.            35.
C                                                                       36.
C     THE VECTORS X AND Y ARE INPUT PARAMETERS IN BOTH CASES.           37.
C     IF MODE = 1, Y SHOULD BE ALTERED WITHOUT CHANGING X.              38.
C     IF MODE = 2, X SHOULD BE ALTERED WITHOUT CHANGING Y.              39.
C     THE PARAMETERS LENIW, LENRW, IW, RW MAY BE USED FOR WORKSPACE     4Ø.
C     AS DESCRIBED BELOW.                                               41.
C                                                                       42.
C     THE RHS VECTOR  B  IS INPUT VIA  U,  AND SUBSEQUENTLY OVERWRITTEN. 43.
C                                                                       44.
C                                                                       45.
C     NOTE.  LSQR USES AN ITERATIVE METHOD TO APPROXIMATE THE SOLUTION. 46.
C     THE NUMBER OF ITERATIONS REQUIRED TO REACH A CERTAIN ACCURACY     47.
C     DEPENDS STRONGLY ON THE SCALING OF THE PROBLEM.  POOR SCALING OF  48.
C     THE ROWS OR COLUMNS OF  A  SHOULD THEREFORE BE AVOIDED WHERE      49.
C     POSSIBLE.                                                         5Ø.
C                                                                       51.
C     FOR EXAMPLE, IN PROBLEM 1 THE SOLUTION IS UNALTERED BY            52.
C     ROW-SCALING.  IF A ROW OF  A  IS VERY SMALL OR LARGE COMPARED TO  53.
C     THE OTHER ROWS OF  A,  THE CORRESPONDING ROW OF  ( A  B )  SHOULD 54.
C     BE SCALED UP OR DOWN.                                             55.
C                                                                       56.
C     IN PROBLEMS 1 AND 2, THE SOLUTION  X  IS EASILY RECOVERED         57.
C     FOLLOWING COLUMN-SCALING.  IN THE ABSENCE OF BETTER INFORMATION,  58.
C     THE NONZERO COLUMNS OF  A  SHOULD BE SCALED SO THAT THEY ALL HAVE 59.
C     THE SAME EUCLIDEAN NORM (E.G.  1.Ø).                              6Ø.
C                                                                       61.
C     IN PROBLEM 3, THERE IS NO FREEDOM TO RE-SCALE IF  DAMP  IS        62.
C     NONZERO.  HOWEVER, THE VALUE OF  DAMP  SHOULD BE ASSIGNED ONLY    63.
C     AFTER ATTENTION HAS BEEN PAID TO THE SCALING OF  A.               64.
C                                                                       65.
C     THE PARAMETER  DAMP  IS INTENDED TO HELP REGULARIZE               66.
C     ILL-CONDITIONED SYSTEMS, BY PREVENTING THE TRUE SOLUTION FROM     67.
C     BEING VERY LARGE.  ANOTHER AID TO REGULARIZATION IS PROVIDED BY   68.
C     THE PARAMETER  ACOND,  WHICH MAY BE USED TO TERMINATE ITERATIONS  69.
C     BEFORE THE COMPUTED SOLUTION BECOMES VERY LARGE.                  7Ø.
C                                                                       71.
C                                                                       72.
C     NOTATION                                                          73.
C     --------                                                          74.
C                                                                       75.
C     THE FOLLOWING QUANTITIES ARE USED IN DISCUSSING THE SUBROUTINE    76.
C     PARAMETERS...                                                     77.
C                                                                       78.
C     ABAR   =  (   A    ),          BBAR  =  ( B )                    79.
C               ( DAMP*I )                   ( Ø )                    8Ø.
C                                                                       81.
C     R      =  B  -  A*X,           RBAR  =  BBAR  -  ABAR*X          82.
C                                                                       83.
C     RNORM  =  SQRT( NORM(R)**2  +  DAMP**2 * NORM(X)**2 )            84.
C            =  NORM( RBAR )                                            85.
C                                                                       86.
C     RELPR  =  THE RELATIVE PRECISION OF FLOATING-POINT ARITHMETIC     87.
C               ON THE MACHINE BEING USED.  FOR EXAMPLE, ON THE IBM 37Ø, 88.
C               RELPR  IS ABOUT 1.ØE-6 AND 1.ØD-16 IN SINGLE AND DOUBLE 89.
C               PRECISION RESPECTIVELY.                                 9Ø.
C                                                                       91.
C     LSQR  MINIMIZES THE FUNCTION  RNORM  WITH RESPECT TO  X.          92.
C                                                                       93.
C                                                                       94.
C     PARAMETERS                                                        95.
C     ----------                                                        96.
C                                                                       97.
C     M       INPUT      THE NUMBER OF ROWS IN  A.                      98.
C                                                                       99.
C     N       INPUT      THE NUMBER OF COLUMNS IN  A.                  1ØØ.
```

```
C                                                                       101.
C     APROD    EXTERNAL   SEE ABOVE.                                   102.
C                                                                       103.
C     DAMP     INPUT      THE DAMPING PARAMETER FOR PROBLEM 3 ABOVE.   104.
C                         (DAMP  SHOULD BE Ø.Ø FOR PROBLEMS 1 AND 2.)  105.
C                         IF THE SYSTEM  A*X = B  IS INCOMPATIBLE, VALUES 106.
C                         OF  DAMP  IN THE RANGE Ø TO SQRT(RELPR)*NORM(A) 107.
C                         WILL PROBABLY HAVE A NEGLIGIBLE EFFECT.      108.
C                         LARGER VALUES OF  DAMP  WILL TEND TO DECREASE 109.
C                         THE NORM OF  X  AND TO REDUCE THE NUMBER OF  110.
C                         ITERATIONS REQUIRED BY LSQR.                 111.
C                                                                      112.
C                         THE WORK PER ITERATION AND THE STORAGE NEEDED 113.
C                         BY LSQR ARE THE SAME FOR ALL VALUES OF  DAMP. 114.
C                                                                      115.
C     LENIW    INPUT      THE LENGTH OF THE WORKSPACE ARRAY  IW.       116.
C     LENRW    INPUT      THE LENGTH OF THE WORKSPACE ARRAY  RW.       117.
C     IW       WORKSPACE  AN INTEGER ARRAY OF LENGTH  LENIW.           118.
C     RW       WORKSPACE  A REAL ARRAY OF LENGTH  LENRW.               119.
C                                                                      120.
C              NOTE.  LSQR DOES NOT EXPLICITLY USE THE PREVIOUS FOUR   121.
C              PARAMETERS, BUT PASSES THEM TO SUBROUTINE APROD FOR     122.
C              POSSIBLE USE AS WORKSPACE.  IF APROD DOES NOT NEED      123.
C              IW  OR  RW,  THE VALUES  LENIW = 1  OR  LENRW = 1  SHOULD 124.
C              BE USED, AND THE ACTUAL PARAMETERS CORRESPONDING TO     125.
C              IW  OR  RW  MAY BE ANY CONVENIENT ARRAY OF SUITABLE TYPE. 126.
C                                                                      127.
C     U(M)     INPUT      THE RHS VECTOR  B.  BEWARE THAT  U  IS       128.
C                         OVER-WRITTEN BY LSQR.                        129.
C                                                                      130.
C     V(N)     WORKSPACE                                               131.
C     W(N)     WORKSPACE                                               132.
C                                                                      133.
C     X(N)     OUTPUT     RETURNS THE COMPUTED SOLUTION  X.            134.
C                                                                      135.
C     SE(N)    OUTPUT     RETURNS STANDARD ERROR ESTIMATES FOR THE     136.
C                         COMPONENTS OF  X.  FOR EACH  I,  SE(I)  IS SET 137.
C                         TO THE VALUE  RNORM * SQRT( SIGMA(I,I) / T ),  138.
C                         WHERE  SIGMA(I,I)  IS AN ESTIMATE OF THE I-TH  139.
C                         DIAGONAL OF THE INVERSE OF ABAR(TRANSPOSE)*ABAR 140.
C                         AND  T = 1       IF  M .LE. N,               141.
C                              T = M - N   IF  M .GT. N  AND  DAMP = Ø,  142.
C                              T = M       IF  DAMP .NE. Ø.              143.
C                                                                      144.
C     ATOL     INPUT      AN ESTIMATE OF THE RELATIVE ERROR IN THE DATA 145.
C                         DEFINING THE MATRIX  A.  FOR EXAMPLE,        146.
C                         IF  A  IS ACCURATE TO ABOUT 6 DIGITS, SET    147.
C                         ATOL = 1.ØE-6 .                              148.
C                                                                      149.
C     BTOL     INPUT      AN ESTIMATE OF THE RELATIVE ERROR IN THE DATA 150.
C                         DEFINING THE RHS VECTOR  B.  FOR EXAMPLE,    151.
C                         IF  B  IS ACCURATE TO ABOUT 6 DIGITS, SET    152.
C                         BTOL = 1.ØE-6 .                              153.
C                                                                      154.
C     CONLIM   INPUT      AN UPPER LIMIT ON  COND(ABAR),  THE APPARENT  155.
C                         CONDITION NUMBER OF THE MATRIX  ABAR.        156.
C                         ITERATIONS WILL BE TERMINATED IF A COMPUTED  157.
C                         ESTIMATE OF  COND(ABAR)  EXCEEDS  CONLIM.    158.
C                         THIS IS INTENDED TO PREVENT CERTAIN SMALL OR 159.
C                         ZERO SINGULAR VALUES OF  A  OR  ABAR  FROM   160.
C                         COMING INTO EFFECT AND CAUSING UNWANTED GROWTH 161.
C                         IN THE COMPUTED SOLUTION.                    162.
C                                                                      163.
C                         CONLIM  AND  DAMP  MAY BE USED SEPARATELY OR 164.
C                         TOGETHER TO REGULARIZE ILL-CONDITIONED SYSTEMS. 165.
C                                                                      166.
C                         NORMALLY,  CONLIM  SHOULD BE IN THE RANGE    167.
C                         1ØØØ  TO  1/RELPR.                           168.
C                         SUGGESTED VALUE --                           169.
C                         CONLIM = 1/(1ØØ*RELPR)  FOR COMPATIBLE SYSTEMS, 170.
C                         CONLIM = 1/(1Ø*SQRT(RELPR))  FOR LEAST SQUARES. 171.
C                                                                      172.
```

```
C            NOTE.  IF THE USER IS NOT CONCERNED ABOUT THE PARAMETERS   173.
C            ATOL, BTOL  AND  CONLIM,  ANY OR ALL OF THEM MAY BE SET    174.
C            TO ZERO.  THE EFFECT WILL BE THE SAME AS THE VALUES        175.
C            RELPR, RELPR  AND  1/RELPR  RESPECTIVELY.                  176.
C                                                                       177.
C     ITNLIM  INPUT      AN UPPER LIMIT ON THE NUMBER OF ITERATIONS.     178.
C                        SUGGESTED VALUE --                             179.
C                        ITNLIM = N/2      FOR WELL CONDITIONED SYSTEMS, 18Ø.
C                        ITNLIM = 4*N      OTHERWISE.                   181.
C                                                                       182.
C     NOUT    INPUT      FILE NUMBER FOR PRINTER.  IF POSITIVE,         183.
C                        A SUMMARY WILL BE PRINTED ON FILE  NOUT.       184.
C                                                                       185.
C     ISTOP   OUTPUT     AN INTEGER GIVING THE REASON FOR TERMINATION... 186.
C                                                                       187.
C               Ø        X = Ø  IS THE EXACT SOLUTION.                  188.
C                        NO ITERATIONS WERE PERFORMED.                  189.
C                                                                       19Ø.
C               1        THE EQUATIONS  A*X = B  ARE PROBABLY           191.
C                        COMPATIBLE.  NORM(A*X - B)  IS SUFFICIENTLY    192.
C                        SMALL, GIVEN THE VALUES OF  ATOL  AND  BTOL.   193.
C                                                                       194.
C               2        THE SYSTEM  A*X = B  IS PROBABLY NOT           195.
C                        COMPATIBLE.  A LEAST-SQUARES SOLUTION HAS      196.
C                        BEEN OBTAINED WHICH IS SUFFICIENTLY ACCURATE,  197.
C                        GIVEN THE VALUE OF  ATOL.                      198.
C                                                                       199.
C               3        AN ESTIMATE OF  COND(ABAR)  HAS EXCEEDED       2ØØ.
C                        CONLIM.  THE SYSTEM  A*X = B  APPEARS TO BE    2Ø1.
C                        ILL-CONDITIONED.  OTHERWISE, THERE COULD BE AN 2Ø2.
C                        AN ERROR IN SUBROUTINE APROD.                  2Ø3.
C                                                                       2Ø4.
C               4        THE EQUATIONS  A*X = B  ARE PROBABLY           2Ø5.
C                        COMPATIBLE.  NORM(A*X - B)  IS AS SMALL AS     2Ø6.
C                        SEEMS REASONABLE ON THIS MACHINE.              2Ø7.
C                                                                       2Ø8.
C               5        THE SYSTEM  A*X = B  IS PROBABLY NOT           2Ø9.
C                        COMPATIBLE.  A LEAST-SQUARES SOLUTION HAS      21Ø.
C                        BEEN OBTAINED WHICH IS AS ACCURATE AS SEEMS    211.
C                        REASONABLE ON THIS MACHINE.                    212.
C                                                                       213.
C               6        COND(ABAR)  SEEMS TO BE SO LARGE THAT THERE IS 214.
C                        NOT MUCH POINT IN DOING FURTHER ITERATIONS,    215.
C                        GIVEN THE PRECISION OF THIS MACHINE.           216.
C                        THERE COULD BE AN ERROR IN SUBROUTINE APROD.   217.
C                                                                       218.
C               7        THE ITERATION LIMIT  ITNLIM  WAS REACHED.      219.
C                                                                       22Ø.
C     ANORM   OUTPUT     AN ESTIMATE OF THE FROBENIUS NORM OF  ABAR.    221.
C                        THIS IS THE SQUARE-ROOT OF THE SUM OF SQUARES  222.
C                        OF THE ELEMENTS OF  ABAR.                      223.
C                        IF  DAMP  IS SMALL AND IF THE COLUMNS OF  A    224.
C                        HAVE ALL BEEN SCALED TO HAVE LENGTH  1.Ø,      225.
C                        ANORM  SHOULD INCREASE TO ROUGHLY  SQRT(N).    226.
C                        A RADICALLY DIFFERENT VALUE FOR  ANORM  MAY    227.
C                        INDICATE AN ERROR IN SUBROUTINE APROD (THERE   228.
C                        MAY BE AN INCONSISTENCY BETWEEN MODES 1 AND 2). 229.
C                                                                       23Ø.
C     ACOND   OUTPUT     AN ESTIMATE OF  COND(ABAR),  THE CONDITION     231.
C                        NUMBER OF  ABAR.  A VERY HIGH VALUE OF  ACOND  232.
C                        MAY AGAIN INDICATE AN ERROR IN APROD.          233.
C                                                                       234.
C     RNORM   OUTPUT     AN ESTIMATE OF THE FINAL VALUE OF NORM(RBAR),  235.
C                        THE FUNCTION BEING MINIMIZED (SEE NOTATION     236.
C                        ABOVE).  THIS WILL BE SMALL IF  A*X = B  HAS   237.
C                        A SOLUTION.                                    238.
C                                                                       239.
C     ARNORM  OUTPUT     AN ESTIMATE OF THE FINAL VALUE OF              24Ø.
C                        NORM( ABAR(TRANSPOSE)*RBAR ), THE NORM OF      241.
C                        THE RESIDUAL FOR THE USUAL NORMAL EQUATIONS.   242.
C                        THIS SHOULD BE SMALL IN ALL CASES.  (ARNORM    243.
C                        WILL OFTEN BE SMALLER THAN THE TRUE VALUE      244.
```

```
C                       COMPUTED FROM THE OUTPUT VECTOR  X.)                245.
C                                                                           246.
C     XNORM   OUTPUT    AN ESTIMATE OF THE NORM OF THE FINAL                247.
C                       SOLUTION VECTOR  X.                                 248.
C                                                                           249.
C                                                                           25Ø.
C     SUBROUTINES AND FUNCTIONS USED                                        251.
C     ------------------------------                                        252.
C                                                                           253.
C     USER        APROD                                                   254.
C     LSQR        NORMLZ                                                  255.
C     BLAS        SCOPY,SNRM2,SSCAL  (SEE LAWSON ET AL. BELOW)            256.
C                 (SNRM2 IS USED ONLY IN NORMLZ)                          257.
C     FORTRAN     ABS,MOD,SQRT                                            258.
C                                                                           259.
C                                                                           26Ø.
C     PRECISION                                                             261.
C     ---------                                                             262.
C                                                                           263.
C     THE NUMBER OF ITERATIONS REQUIRED BY LSQR WILL USUALLY DECREASE       264.
C     IF THE COMPUTATION IS PERFORMED IN HIGHER PRECISION.  TO CONVERT      265.
C     LSQR  AND  NORMLZ  BETWEEN SINGLE- AND DOUBLE-PRECISION, CHANGE      266.
C     THE WORDS                                                             267.
C                 SCOPY, SNRM2, SSCAL                                       268.
C                 ABS, REAL, SQRT                                           269.
C     TO THE APPROPRIATE BLAS AND FORTRAN EQUIVALENTS.                      27Ø.
C                                                                           271.
C                                                                           272.
C     REFERENCES                                                            273.
C     ----------                                                            274.
C                                                                           275.
C     PAIGE, C.C. AND SAUNDERS, M.A.  LSQR: AN ALGORITHM FOR SPARSE         276.
C        LINEAR EQUATIONS AND SPARSE LEAST SQUARES.                         277.
C        ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE 8, 1 (MARCH 1982).       278.
C                                                                           279.
C     LAWSON, C.L., HANSON, R.J., KINCAID, D.R. AND KROGH, F.T.             28Ø.
C        BASIC LINEAR ALGEBRA SUBPROGRAMS FOR FORTRAN USAGE.                281.
C        ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE 5, 3 (SEPT 1979),        282.
C        3Ø8-323 AND 324-325.                                               283.
C                                                                           284.
C                                                                           285.
C     LSQR.       THIS VERSION DATED 22 FEBRUARY 1982.                      286.
C     ------------------------------------------------------------------    287.
```

ALGORITHM 584
CUBTRI: Automatic Cubature over a Triangle

D. P. LAURIE

National Research Institute for Mathematical Sciences, South Africa

Categories and Subject Descriptors: G.1.4 [**Numerical Analysis**]: Quadrature and Numerical Differentiation—*adaptive quadrature, multiple quadrature*; G.m [**Mathematics of Computing**]: Miscellaneous—*FORTRAN*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Quadrature rule

DESCRIPTION AND PURPOSE

Given a triangle T with vertices (t_{1j}, t_{2j}), $j = 1, 2, 3$, a function $f(x, y)$ defined on T, and a tolerance $\epsilon > 0$, CUBTRI attempts to compute a number A such that

$$\left| A - \int\int_T f(x, y)\, dx\, dy \right| < \max\{\epsilon, |A\epsilon|\}. \tag{1}$$

The method employed is similar to that of Haegemans [1] and consists of the following steps.

(1) Compute an approximate integral A and error estimate η for the original triangle T.
(2) If the current values of A and η satisfy $\eta < \max\{\epsilon, |A\epsilon|\}$, or if further computation will exceed specified limits, exit.
(3) Given a list of triangles whose union is T, together with an approximate integral and error estimate for each triangle, identify the triangle T_k with largest error estimate.
(4) Remove T_k from the data list and subtract its contributions from the current values of A and η.
(5) Divide T_k into four congruent triangles, append them to the data list, and compute an approximate integral and error estimate for each. Add these contributions to A and η.
(6) Diagnose whether round-off errors are contaminating the obtained values to a point where further subdivision is futile. If so, exit; else return to step 2.

The general strategy falls within the framework outlined by Rice [7], apart from the test for roundoff, which is due to Piessens [5] and also appears in Haegemans' routine TRIADA [1].

Received 28 August 1979; revised 27 January 1982; accepted 28 January 1982
Author's address: National Research Institute for the Mathematical Sciences of CSIR, P.O. Box 395, Pretoria, South Africa.

Table I. The Integration Formula Pair Used by CUBTRI

i	$\zeta_1^{(i)}$	$\zeta_2^{(i)}$	$\zeta_3^{(i)}$	$w_1^{(i)}$	$w_2^{(i)}$
0	1/3	1/3	1/3	9/40	7137/62720 $-45\sigma/1568$
1, 2	3/7 $\pm 2\phi/21$	2/7 $\mp\phi/21$	$=\zeta_2^{(i)}$	31/80 $\mp\phi/400$	$-9301697/4695040$ $\mp 13517313\phi/23475200$ $+764885\sigma/939008$ $\pm 198763\phi\sigma/939008$
3, 4	4/9 $\pm\phi/9$ $+\sigma/9$ $\mp\sigma\phi/45$	5/18 $\mp\phi/18$ $-\sigma/18$ $\pm\sigma\phi/90$	$=\zeta_2^{(i)}$	0	102791225/59157504 $\pm 23876225\phi/59157504$ $-34500875\sigma/59157504$ $\mp 9914825\phi\sigma/59157504$
5	$=\zeta_2^{(3)}$	$\zeta_2^{(4)}$	4/9 $+\sigma/9$	0	11075/8064 $-125\sigma/288$

$\phi = \sqrt{15}$, $\sigma = \sqrt{7}$

$$Q^{(i)}f = \frac{1}{6}\sum_{\substack{k=1\\k\neq l}}^{3}\sum_{l=1}^{3} f(\zeta_k^{(i)}, \zeta_l^{(i)})$$

$$Q_1 f = \Delta \sum_{i=0}^{2} w_1^{(i)} Q^{(i)} f$$

$$Q_2 f = \Delta \sum_{i=0}^{5} w_2^{(i)} Q^{(i)} f$$

Δ = area of triangle

$\zeta_k^{(i)}$ are barycentric coordinates

CUBTRI attempts to improve on TRIADA in terms of number of function evaluations and storage management.

NUMBER OF FUNCTION EVALUATIONS

Two cubature rules Q_1 and Q_2 are applied to each triangle to produce estimates A_1 and A_2, where A_2 is of a higher degree than A_1. This gives an approximate integral $A = A_2$ and error estimate $\eta = |A_2 - A_1|$ which is likely to be pessimistic, since it really estimates the error in A_1 rather than that in A_2. This effect is countered by employing a device given in [3] for sharpening the error estimate. It is, of course, possible to use two completely independent cubature rules (e.g., Haegemans [1] uses conic product Gauss formulas (see Stroud [8], Section 2.5) with 36 and 49 points, respectively), but it seems sensible to extend Kronrod's [2] idea, that Q_2 should reuse the points of Q_1, to the two-dimensional case.

In CUBTRI, Q_1 is the 7-point degree-5 rule of Radon [6], and Q_2 is a new 19-point degree-8 rule given in Table I. The theory of Lyness and Jespersen [4] shows that 19 points and degree 8 are optimal, given that Q_2 has D_3 symmetry and contains the 7 points of Q_1. A curious feature of Q_2 is that the 12 new points are the intersection points of three pairs of parallel lines (see Figure 1).

STORAGE MANAGEMENT

Most of the storage requirements arise from the need to store a list of triangles. In general, 6 numbers are necessary to describe a triangle (e.g., three vertices of two coordinates each), giving a total of $6n$ numbers for n triangles.

Haegemans [1] has improved on this by keeping a separate list of vertices; each triangle can then be described by 3 integers. The length of the list is only $n + 2$, because each subdivision creates as many new vertices (three) as the net increase in the number of triangles. This yields a total of $3n$ integers and $2n + 4$ reals.

In CUBTRI we take advantage of the fact that the sides of each subtriangle are parallel to those of the original triangle. Geometrically, one can visualize the subtriangle as being obtained by translation, scaling, and possibly rotation

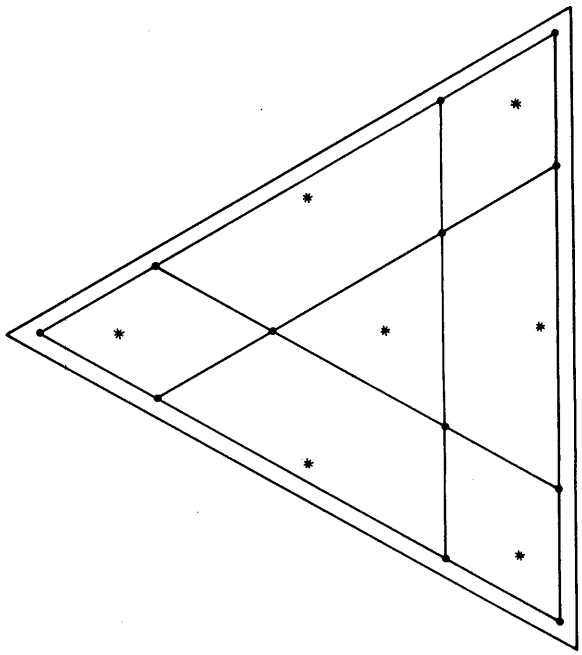


Fig. 1. Disposition of points in the degree-8 formula. •—Additional points of degree-8 formula. *—Points of Radon's degree-5 formula.

through 180°. These effects can be represented by two coordinates, a scale factor, and the sign of the scale factor, respectively: thus three numbers per triangle suffice. It is convenient to use barycentric coordinates for the translation vector, because in that case only numbers that are terminating binary fractions are generated; these are exact machine numbers (even in single precision) on most computers, and rounding errors are avoided. Only $3n$ reals are therefore required in addition to the vertices of T.

TESTS

The driver main program and function $f(x, y)$ supplied with the algorithm calculate the following seven integrals.

Case 1.

$$\int_0^1 \int_0^x (x^2 + 3y^2)^{-1/2}\, dy\, dx = \frac{\log(2 + \sqrt{3})}{\sqrt{3}}.$$

This integrand has a singularity at the origin.

Case 2. The same as Case 1, with a rapidly oscillatory relative error of magnitude 10^{-6} perturbing the integrand. This simulates the behavior of a machine with a short word length.

Case 3.

$$\int_0^1 \int_0^x f(x, y)\, dy\, dx = \frac{\pi}{8}$$

where

$$f(x, y) = \begin{cases} 1, & (x - \frac{1}{2})^2 + (y - \frac{1}{2})^2 \le \frac{1}{2}, \\ 0, & \text{otherwise.} \end{cases}$$

This integrand has a jump discontinuity along a curved line, the circle of radius $\frac{1}{2}$ centered at $(\frac{1}{2}, \frac{1}{2})$.

The following four test functions were suggested by Haegemans [1].

Case 4.

$$\int_0^1 \int_0^{1-x} \exp(\sin x \cos y)\, dy\, dx \approx 0.6918104506612316.$$

Table II. $f(x, y) = (x^2 + 3y^2)^{-1/2}$

	CUBTRI				TRIADA			
ϵ	NFE	EST	ERR	IER	NFE	EST	ERR	IER
1_{10-3}	323	6.2_{10-4}	6.1_{10-4}	0	765	9.1_{10-4}	3.0_{10-3}	0
1_{10-4}	551	8.5_{10-5}	7.5_{10-5}	0	2125	5.7_{10-5}	1.9_{10-4}	0
1_{10-5}	1007	8.1_{10-6}	1.5_{10-6}	0	3145	7.1_{10-6}	2.3_{10-5}	0
1_{10-6}	1919	8.6_{10-7}	3.7_{10-8}	0	4165	9.0_{10-7}	2.9_{10-6}	0
1_{10-7}	3135	9.9_{10-8}	3.0_{10-9}	0	5525	6.3_{10-8}	1.8_{10-7}	0
1_{10-8}	5035	9.5_{10-9}	1.2_{10-10}	0	7225	9.5_{10-9}	6.6_{10-9}	0
1_{10-9}	7391	9.8_{10-10}	7.6_{10-11}	0	10285	9.8_{10-10}	9.8_{10-10}	0
						10		

Table III. $f(x, y) = (x^2 + 3y^2)^{-1/2}(1 + \eta), |\eta| \leq 10^{-6}$

	CUBTRI				TRIADA			
ϵ	NFE	EST	ERR	IER	NFE	EST	ERR	IER
1_{10-3}	323	6.2_{10-4}	6.1_{10-4}	0	765	9.1_{10-4}	3.0_{10-3}	0
1_{10-4}	551	8.5_{10-5}	7.5_{10-5}	0	2125	5.7_{10-5}	1.9_{10-4}	0
1_{10-5}	1007	8.2_{10-6}	1.5_{10-6}	0	3145	7.2_{10-6}	2.3_{10-5}	0
1_{10-6}	1919	8.6_{10-7}	2.8_{10-8}	0	4165	9.7_{10-7}	2.9_{10-6}	0
1_{10-7}	3135	9.8_{10-8}	1.1_{10-8}	0	6205	1.0_{10-7}	8.9_{10-9}	0
1_{10-8}	5415	1.3_{10-8}	1.5_{10-8}	4	9945	6.5_{10-8}	1.7_{10-10}	2

Table IV. $f(x, y) = \frac{1}{2} + \frac{1}{2}\operatorname{sign}\left\{\frac{1}{4} - \left(x - \frac{1}{2}\right)^2 - \left(y - \frac{1}{2}\right)^2\right\}$

	CUBTRI				TRIADA			
ϵ	NFE	EST	ERR	IER	NFE	EST	ERR	IER
1_{10-3}	551	3.9_{10-3}	2.6_{10-3}	4	10625	9.9_{10-4}	6.4_{10-5}	0
1_{10-4}	3059	2.0_{10-3}	5.7_{10-4}	4	14705	9.3_{10-4}	3.0_{10-4}	1

This integral seems to be impossible to evaluate analytically; the numerical "exact" value is given in [1].

Case 5.

$$\int_0^1 \int_0^{1-x} \frac{x^2}{(1 + x^2)}\, dy\, dx = \frac{1 + \log 2}{2} - \frac{\pi}{4}.$$

The integrand depends on x only.

Case 6.

$$\int_0^1 \int_0^{1-x} \sin(3x + 6y)\, dy\, dx = \frac{1}{9}\sin 3 - \frac{1}{18}\sin 6.$$

A mildly oscillating integrand.

Case 7.

$$\int_0^1 \int_0^{1-x} \sin(x + y)^{-1/2}\, dy\, dx = \frac{2}{3}.$$

The integrand has a gentler singularity at the origin than that of Case 1.

In Tables II–VIII we report the results obtained by CUBTRI as well as by TRIADA [2] on a CDC Cyber 174 (48-bit mantissa, binary floating point). The column legends are

ϵ Required tolerance.
NFE Number of function evaluations.
EST Estimated absolute error.
ERR Actual absolute error.

Table V. $f(x, y) = \exp(\sin x \cos y)$

	CUBTRI				TRIADA			
ϵ	NFE	EST	ERR	IER	NFE	EST	ERR	IER
$1_{10^{-3}}$	19	$1.8_{10^{-5}}$	$2.6_{10^{-9}}$	0	85	$1.5_{10^{-10}}$	$2.3_{10^{-12}}$	0
$1_{10^{-5}}$	95	$4.0_{10^{-11}}$	$4.6_{10^{-12}}$	0	85	$1.5_{10^{-10}}$	$2.3_{10^{-12}}$	0

Table VI. $f(x, y) = x^2/(1 + x^2)$

	CUBTRI				TRIADA			
ϵ	NFE	EST	ERR	IER	NFE	EST	ERR	IER
$1_{10^{-3}}$	19	$3.2_{10^{-5}}$	$1.7_{10^{-7}}$	0	85	$8.6_{10^{-10}}$	$6.0_{10^{-11}}$	0
$1_{10^{-5}}$	95	$7.2_{10^{-10}}$	$3.6_{10^{-10}}$	0	85	$8.6_{10^{-10}}$	$6.0_{10^{-11}}$	0

Table VII. $f(x, y) = \sin(3x + 6y)$

	CUBTRI				TRIADA			
ϵ	NFE	EST	ERR	IER	NFE	EST	ERR	IER
$1_{10^{-3}}$	19	$3.3_{10^{-4}}$	$2.1_{10^{-8}}$	0	85	$1.9_{10^{-9}}$	$2.1_{10^{-11}}$	0
$1_{10^{-4}}$	95	$1.6_{10^{-9}}$	$1.3_{10^{-11}}$	0	85	$1.9_{10^{-9}}$	$2.1_{10^{-11}}$	0
$1_{10^{-9}}$	171	$8.7_{10^{-10}}$	$2.9_{10^{-11}}$	0	425	$2.4_{10^{-12}}$	$4.8_{10^{-15}}$	0

Table VIII. $f(x + y) = (x + y)^{-1/2}$

	CUBTRI				TRIADA			
ϵ	NFE	EST	ERR	IER	NFE	EST	ERR	IER
$1_{10^{-3}}$	95	$1.4_{10^{-4}}$	$1.4_{10^{-4}}$	0	85	$3.2_{10^{-4}}$	$6.2_{10^{-4}}$	0
$1_{10^{-4}}$	171	$4.9_{10^{-5}}$	$4.9_{10^{-5}}$	0	765	$4.8_{10^{-5}}$	$9.8_{10^{-5}}$	0
$1_{10^{-5}}$	323	$6.3_{10^{-6}}$	$6.1_{10^{-6}}$	0	1445	$6.0_{10^{-6}}$	$1.2_{10^{-5}}$	0
$1_{10^{-6}}$	475	$9.4_{10^{-7}}$	$7.7_{10^{-7}}$	0	2125	$7.6_{10^{-7}}$	$1.5_{10^{-6}}$	0
$1_{10^{-7}}$	1007	$7.2_{10^{-8}}$	$1.2_{10^{-8}}$	0	2805	$9.5_{10^{-8}}$	$1.9_{10^{-7}}$	0
$1_{10^{-8}}$	1615	$9.1_{10^{-9}}$	$1.5_{10^{-9}}$	0	3825	$4.2_{10^{-9}}$	$8.4_{10^{-9}}$	0
$1_{10^{-9}}$	2375	$8.8_{10^{-10}}$	$5.6_{10^{-11}}$	0	4505	$5.5_{10^{-10}}$	$1.1_{10^{-9}}$	0

IER Error flag. For TRIADA, IER = 1 denotes that the limit on NFE has been reached, and IER = 2 that the round-off error threshold has been reached. For CUBTRI, the meaning of IER is explained in the leading comments.

We can draw the following conclusions.

(1) *Reliability.* The error estimates produced by CUBTRI after subdivision range from 0.87 to 120 times the actual error, and in only one instance is the actual error underestimated. Before subdivision (i.e., when NFE = 19), the device [3] is unavailable, and the overestimation is much more pessimistic. The error estimates produced by TRIADA range from 0.3 to 500 times the actual error, and in 16 instances (all involving nonsmooth integrands) the actual error is underestimated.

(2) *Accuracy.* On smooth integrands TRIADA is slightly more accurate than CUBTRI (e.g., in Case 2, TRIADA with NFE = 85 produces ERR = 2.3_{10}^{-12}). On nonsmooth integrands CUBTRI is much more accurate than TRIADA (e.g., in Case 1, CUBTRI with NFE = 1919 produces ERR = 3.7_{10}^{-8}, whereas TRIADA with NFE = 2125 can only obtain ERR = 1.9_{10}^{-4}).

(3) *Efficiency.* On smooth integrands TRIADA in 15 instances out of 21 terminated at a given ϵ with fewer function evaluations than CUBTRI. On nonsmooth integrands, in 20 instances out of 21 CUBTRI terminated with fewer function evaluations than TRIADA.

On machines with a short word length, termination with IER = 4 can be expected when ϵ is smaller than machine accuracy, and the behavior of the algorithm when ϵ is close to machine accuracy might be somewhat different. The situation is simulated by the difference between Cases 1 and 2.

To summarize, CUBTRI is a reliable and efficient integrator on smooth as well as nonsmooth integrands, although not quite so accurate and efficient for smooth integrands as TRIADA.

ACKNOWLEDGMENTS

The referee of an early version of this paper drew my attention to [4]. Elsie de Lange wrote the driver main program and test functions. Ina Marais checked the correctness of Table I.

REFERENCES

1. HAEGEMANS, A. An algorithm for the automatic integration over a triangle. *Comput. 19* (1977), Springer-Verlag, New York, pp. 179–187.
2. KRONROD, A.S. *Nodes and Weights of Gaussian Quadrature Formulas.* Consultants Bureau, New York, 1965.
3. LAURIE, D.P. Sharper error estimates in adaptive numerical integration. NRIMS Tech. Rep. TWISK 259, National Research Institute for the Mathematical Science, Pretoria, South Africa, 1982.
4. LYNESS, J.N., AND JESPERSEN, D. Moderate degree symmetric quadrature rules for the triangle. *J. Inst. Math. Appl. 15* (1975), 19–32.
5. PIESSENS, R. A quadrature routine with roundoff error guard. Rep. TW 17, Applied Mathematics and Programming Division, Univ. Leuven, Leuven, Belgium, 1974.
6. RADON, J. Zur mechanische Kubatur. *Monatsh. Math. 52* (1948), 286–300.
7. RICE, J.R. A metalgorithm for adaptive quadrature. *J. ACM 22*, 1(Jan. 1975), 61–82.
8. STROUD, A.H. *Approximate Calculation of Multiple Integrals.* Prentice-Hall, Englewood Cliffs, N.J., 1971.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Distribution Service].

```
      SUBROUTINE CUBTRI(F, T, EPS, MCALLS, ANS, ERR, NCALLS, W, NW,     CUB   1Ø
     * IDATA, RDATA, IER)                                               CUB   2Ø
C                                                                       CUB   3Ø
C       ADAPTIVE CUBATURE OVER A TRIANGLE                               CUB   4Ø
C                                                                       CUB   5Ø
C       PARAMETERS                                                      CUB   6Ø
C          F      - USER SUPPLIED EXTERNAL FUNCTION OF THE FORM         CUB   7Ø
C                   F(X,Y,IDATA,RDATA)                                  CUB   8Ø
C                   WHERE X AND Y ARE THE CARTESIAN COORDINATES OF A    CUB   9Ø
C                   POINT IN THE PLANE, AND IDATA AND RDATA ARE INTEGER CUB  1ØØ
C                   AND REAL VECTORS IN WHICH DATA MAY BE PASSED.       CUB  11Ø
C          T      - ARRAY OF DIMENSION (2,3) WHERE T(1,J) AND T(2,J)    CUB  12Ø
C                   ARE THE X AND Y COORDINATES OF THE J-TH VERTEX OF   CUB  13Ø
C                   THE GIVEN TRIANGLE (INPUT)                          CUB  14Ø
C          EPS    - REQUIRED TOLERANCE (INPUT).  IF THE COMPUTED        CUB  15Ø
C                   INTEGRAL IS BETWEEN-1 AND 1, AN ABSOLUTE ERROR      CUB  16Ø
C                   TEST IS USED, ELSE A RELATIVE ERROR TEST IS USED.   CUB  17Ø
C          MCALLS- MAXIMUM PERMITTED NUMBER OF CALLS TO F (INPUT)       CUB  18Ø
C          ANS    - ESTIMATE FOR THE VALUE OF THE INTEGRAL OF F OVER    CUB  19Ø
C                   THE GIVEN TRIANGLE (OUTPUT)                         CUB  2ØØ
C          ERR    - ESTIMATED ABSOLUTE ERROR IN ANS (OUTPUT)            CUB  21Ø
C          NCALLS- ACTUAL NUMBER OF CALLS TO F (OUTPUT).  THIS          CUB  22Ø
C                   PARAMETER MUST BE INITIALIZED TO Ø ON THE FIRST     CUB  23Ø
C                   CALL TO CUBTRI FOR A GIVEN INTEGRAL (INPUT)         CUB  24Ø
C          W      - WORK SPACE.  MAY NOT BE DESTROYED BETWEEN CALLS TO  CUB  25Ø
C                   CUBTRI IF RESTARTING IS INTENDED                    CUB  26Ø
C          NW     - LENGTH OF WORK SPACE (INPUT).                       CUB  27Ø
C                   IF NW .GE. 3*(19+3*MCALLS)/38, TERMINATION DUE TO   CUB  28Ø
C                   FULL WORK SPACE WILL NOT OCCUR.                     CUB  29Ø
C          IER    - TERMINATION INDICATOR (OUTPUT)                      CUB  3ØØ
```

```
C                IER=Ø   NORMAL TERMINATION, TOLERANCE SATISFIED      CUB  31Ø
C                IER=1   MAXIMUM NUMBER OF CALLS REACHED              CUB  32Ø
C                IER=2   WORK SPACE FULL                              CUB  33Ø
C                IER=3   FURTHER SUBDIVISION OF TRIANGLES IMPOSSIBLE  CUB  34Ø
C                IER=4   NO FURTHER IMPROVEMENT IN ACCURACY IS        CUB  35Ø
C                      POSSIBLE DUE TO ROUNDING ERRORS IN FUNCTION    CUB  36Ø
C                      VALUES                                         CUB  37Ø
C                IER=5   NO FURTHER IMPROVEMENT IN ACCURACY IS        CUB  38Ø
C                      POSSIBLE BECAUSE SUBDIVISION DOES NOT          CUB  39Ø
C                      CHANGE THE ESTIMATED INTEGRAL. MACHINE         CUB  4ØØ
C                      ACCURACY HAS PROBABLY BEEN REACHED BUT         CUB  41Ø
C                      THE ERROR ESTIMATE IS NOT SHARP ENOUGH.        CUB  42Ø
C                                                                     CUB  43Ø
C      CUBTRI IS DESIGNED TO BE CALLED REPEATEDLY WITHOUT WASTING     CUB  44Ø
C      EARLIER WORK.  THE PARAMETER NCALLS IS USED TO INDICATE TO     CUB  45Ø
C      CUBTRI AT WHAT POINT TO RESTART, AND MUST BE RE-INITIALIZED    CUB  46Ø
C      TO Ø WHEN A NEW INTEGRAL IS TO BE COMPUTED.  AT LEAST ONE OF   CUB  47Ø
C      THE PARAMETERS EPS, MCALLS AND NW MUST BE CHANGED BETWEEN      CUB  48Ø
C      CALLS TO CUBTRI, ACCORDING TO THE RETURNED VALUE OF IER. NONE  CUB  49Ø
C      OF THE OTHER PARAMETERS MAY BE CHANGED IF RESTARTING IS DONE.  CUB  5ØØ
C      IF IER=3 IS ENCOUNTERED, THERE PROBABLY IS A SINGULARITY       CUB  51Ø
C      SOMEWHERE IN THE REGION.  THE ERROR MESSAGE INDICATES THAT     CUB  52Ø
C      FURTHER SUBDIVISION IS IMPOSSIBLE BECAUSE THE VERTICES OF THE  CUB  53Ø
C      SMALLER TRIANGLES PRODUCED WILL BEGIN TO COALESCE TO THE       CUB  54Ø
C      PRECISION OF THE COMPUTER.  THIS SITUATION CAN USUALLY BE      CUB  55Ø
C      RELIEVED BY SPECIFYING THE REGION IN SUCH A WAY THAT THE       CUB  56Ø
C      SINGULARITY IS LOCATED AT THE THIRD VERTEX OF THE TRIANGLE.    CUB  57Ø
C      IF IER=4 IS ENCOUNTERED, THE VALUE OF THE INTEGRAL CANNOT BE   CUB  58Ø
C      IMPROVED ANY FURTHER. THE ONLY EXCEPTION TO THIS OCCURS WHEN A CUB  59Ø
C      FUNCTION WITH HIGHLY IRREGULAR BEHAVIOUR IS INTEGRATED (E.G.   CUB  6ØØ
C      FUNCTIONS WITH JUMP DISCONTINUITIES OR VERY HIGHLY OSCILLATORY CUB  61Ø
C      FUNCTIONS). IN SUCH A CASE THE USER CAN DISABLE THE ROUNDING   CUB  62Ø
C      ERROR TEST BY REMOVING THE IF STATEMENT SHORTLY AFTER STATEMENT CUB  63Ø
C      NUMBER 7Ø.                                                     CUB  64Ø
C                                                                     CUB  65Ø

      SUBROUTINE CUBRUL(F, VEC, P, IDATA, RDATA)                      CUB   1Ø
C                                                                     CUB   2Ø
C      BASIC CUBATURE RULE PAIR OVER A TRIANGLE                       CUB   3Ø
C                                                                     CUB   4Ø
C      PARAMETERS                                                     CUB   5Ø
C        F  - EXTERNAL FUNCTION - SEE COMMENTS TO CUBTRI              CUB   6Ø
C        VEC- MATRIX OF BASE VECTORS AND ORIGIN (INPUT)               CUB   7Ø
C        P  - TRIANGLE DESCRIPTION VECTOR OF DIMENSION 6              CUB   8Ø
C             P(1) - TRANSFORMED X COORDINATE OF ORIGIN VERTEX(INPUT) CUB   9Ø
C             P(2) - TRANSFORMED Y COORDINATE OF ORIGIN VERTEX(INPUT) CUB  1ØØ
C             P(3) - DISTANCE OF OTHER VERTICES IN THE DIRECTIONS     CUB  11Ø
C                    OF THE BASE VECTORS (INPUT)                     CUB  12Ø
C             P(4) - LESS ACCURATE ESTIMATED INTEGRAL (OUTPUT)        CUB  13Ø
C             P(5) - MORE ACCURATE ESTIMATED INTEGRAL (OUTPUT)        CUB  14Ø
C             P(6) - ABS(P(5)-P(4))   (OUTPUT)                        CUB  15Ø
C                                                                     CUB  16Ø
C      CUBRUL EVALUATES A LINEAR COMBINATION OF BASIC INTEGRATION     CUB  17Ø
C      RULES HAVING D3 SYMMETRY.  THE AREAL COORDINATES PERTAINING TO CUB  18Ø
C      THE J-TH RULE ARE STORED IN W(I,J),I=1,2,3.  THE CORRESPONDING CUB  19Ø
C      WEIGHTS ARE W(4,J) AND W(5,J), WITH W(5,J) BELONGING TO THE    CUB  2ØØ
C      MORE ACCURATE FORMULA.  IF W(1,J).EQ.W(2,J), THE INTEGRATION   CUB  21Ø
C      POINT IS THE CENTROID, ELSE IF W(2,J).EQ.W(3,J), THE EVALUATION CUB  22Ø
C      POINTS ARE ON THE MEDIANS.  IN BOTH CASES ADVANTAGE IS TAKEN OF CUB  23Ø
C      SYMMETRY TO AVOID REPEATING FUNCTION EVALUATIONS.              CUB  24Ø
C                                                                     CUB  25Ø
C      THE FOLLOWING DOUBLE PRECISION VARIABLES ARE USED TO AVOID     CUB  26Ø
C      UNNECESSARY ROUNDING ERRORS IN FLOATING POINT ADDITION.        CUB  27Ø
C      THEY MAY BE DECLARED SINGLE PRECISION IF DOUBLE PRECISION IS   CUB  28Ø
C      NOT AVAILABLE AND FULL ACCURACY IS NOT NEEDED.                 CUB  29Ø
      DOUBLE PRECISION A1, A2, S, SN, DZERO, DONE, DTHREE, DSIX       CUB  3ØØ
      REAL AREA, ORIGIN(2), P(6), RDATA(1), TVEC(2,3), VEC(2,3), W(5,6) CUB 31Ø
      INTEGER IDATA(1)                                                CUB  32Ø
C                                                                     CUB  33Ø
C      W CONTAINS POINTS AND WEIGHTS OF THE INTEGRATION FORMULAE      CUB  34Ø
C      NQUAD - NUMBER OF BASIC RULES USED                             CUB  35Ø
C                                                                     CUB  36Ø
C      THIS PARTICULAR RULE IS THE 19 POINT EXTENSION (DEGREE 8) OF   CUB  37Ø
```

```
C       THE FAMILIAR 7 POINT RULE (DEGREE 5).                                CUB  38Ø
C                                                                            CUB  39Ø
C     SIGMA=SQRT(7)                                                          CUB  4ØØ
C     PHI=SQRT(15)                                                           CUB  41Ø
C     W(1,1),W(2,1),W(3,1) = 1/3                                             CUB  42Ø
C     W(4,1) = 9/4Ø                                                          CUB  43Ø
C     W(5,1) = 7137/6272Ø - 45*SIGMA/1568                                    CUB  44Ø
C     W(1,2) = 3/7 + 2*PHI/21                                                CUB  45Ø
C     W(2,2),W(3,2) = 2/7 - PHI/21                                           CUB  46Ø
C     W(4,2) = 31/8Ø - PHI/4ØØ                                               CUB  47Ø
C     W(5,2) = - 93Ø1697/4695Ø4Ø - 13517313*PHI/2347525ØØ                    CUB  48Ø
C            + 764885*SIGMA/939ØØ8 + 198763*PHI*SIGMA/939ØØ8                 CUB  49Ø
C     W(*,3) = W(*,2) WITH PHI REPLACED BY -PHI                              CUB  5ØØ
C     W(1,5) = 4/9 + PHI/9 + SIGMA/9 - SIGMA*PHI/45                          CUB  51Ø
C     W(2,5),W(3,5) = 5/18 - PHI/18 - SIGMA/18 + SIGMA*PHI/9Ø                CUB  52Ø
C     W(4,5) = Ø                                                             CUB  53Ø
C     W(5,5) = 1Ø2791225/59157504 + 23876225*PHI/5915750Ø                    CUB  54Ø
C            - 345ØØ875*SIGMA/5915750Ø - 9914825*PHI*SIGMA/5915750Ø          CUB  55Ø
C     W(*,4) = W(*,5) WITH PHI REPLACED BY -PHI                              CUB  56Ø
C     W(1,6) = 4/9 + SIGMA/9                                                 CUB  57Ø
C     W(2,6) = W(2,4)                                                        CUB  58Ø
C     W(3,6) = W(2,5)                                                        CUB  59Ø
C     W(4,6) = Ø                                                             CUB  6ØØ
C     W(5,6) = 11Ø75/8Ø64 - 125*SIGMA/288                                    CUB  61Ø
C                                                                            CUB  62Ø
```

ALGORITHM 585
A Subroutine for the General Interpolation and Extrapolation Problems

C. BREZINSKI
University of Lille, France

Categories and Subject Descriptors: G.m [**Mathematics of Computing**]: Miscellaneous—*FORTRAN*

General Terms: Algorithms

Additional Key Words and Phrases: Convergence acceleration, extrapolation, interpolation, least squares approximation, Neville-Aitken scheme

1. INTRODUCTION

Recently a generalization of Neville-Aitken algorithm for extrapolation has been found by Håvie [10]. This general extrapolation scheme has also been studied by Brezinski [3], who showed that it includes most of the convergence acceleration methods actually known. This algorithm has been called the E-algorithm, or the BH-protocol [15, Chaps. 10 and 11, pp. 175–209].

Of course, after that, the next step was to look for a general interpolation scheme for interpolation by a linear combination of functions forming a Chebyshev system. Such an algorithm was in fact found some years ago by Mühlbach [12, 13]. Except for the initializations, it is the same as the E-algorithm. This algorithm has been called the Mühlbach–Neville-Aitken algorithm (MNA-algorithm). The same algorithm can also be used for the general interpolation problem [4] as described, for example, by Davis [7].

The E-algorithm and its programming are described in Section 2. A FORTRAN subroutine is given with an application. Section 3 is devoted to the MNA-algorithm, while Section 4 deals with least squares approximation and extrapolation.

2. THE E-ALGORITHM

Let (S_n) be a sequence of real numbers and let us consider the sequence transformation defined by

$$E_k(S_n) = \frac{\begin{vmatrix} S_n & S_{n+1} & \cdots & S_{n+k} \\ g_1(n) & g_1(n+1) & \cdots & g_1(n+k) \\ \vdots & \vdots & & \vdots \\ g_k(n) & g_k(n+1) & \cdots & g_k(n+k) \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ g_1(n) & g_1(n+1) & \cdots & g_1(n+k) \\ \vdots & \vdots & & \vdots \\ g_k(n) & g_k(n+1) & \cdots & g_k(n+k) \end{vmatrix}}$$

Received 20 June 1980; revised 30 January 1981; accepted 25 March 1982
This work was partly supported by NATO Research Grant 027-81.
Author's address: Université des Sciences et Techniques de Lille, B.P. 36, 59655 Villeneuve d'Ascq Cedex, France.

where the g_i's are given sequences such that the denominator in $E_k(S_n)$ does not vanish.

The main basic algebraic property of this transformation is that $\forall n \geq N$, $E_k(S_n) = S$ if the sequence (S_n) satisfies, $\forall n \geq N$

$$S_n = S + a_1 g_1(n) + \cdots + a_k g_k(n).$$

This formalism is quite general since it includes most of the sequence transformations actually used to accelerate the convergence of the sequence (S_n). For example, if $g_i(n) = \Delta S_{n+i-1}$, it is Shanks' transformation [14]; if $g_i(n) = x_n^{i-1}\Delta S_n$, it is Levin's method [11]; for $g_i(n) = x_n^i$ we recover Richardson extrapolation process; and for $g_i(n) = (\Delta S_n)^i$, it is Germain-Bonne's method [9]; and so forth.

From the numerical point of view, a recursive method to avoid the computation of the determinants involved in $E_k(S_n)$ is needed. Such algorithms exist for the particular cases quoted above: Wynn's ϵ-algorithm [16] for Shanks' transformation, Neville-Aitken's scheme for Richardson process and Germain-Bonne's transformation, some special devices [5] for Levin's method, and so on. The corresponding FORTRAN subroutines can be found in [1]. Thus we have at least as many different algorithms as sequence transformations.

Recently a recursive algorithm working in the general case has been obtained by Håvie [10] and Brezinski [3]. This algorithm, called the E-algorithm, is as follows:

Initializations $\quad E_0^{(n)} = S_n, \qquad n = 0, 1, \ldots$

$\qquad\qquad\quad g_{0,i}^{(n)} = g_i(n), \qquad i = 1,2, \ldots$ and $n = 0, 1, \ldots$.

Then, for $k = 1, 2, \ldots$ and $n = 0, 1, \ldots$

$$E_k^{(n)} = E_{k-1}^{(n)} + g_{k-1,k}^{(n)} \frac{E_{k-1}^{(n)} - E_{k-1}^{(n+1)}}{g_{k-1,k}^{(n+1)} - g_{k-1,k}^{(n)}} \tag{2.1}$$

$$g_{k,i}^{(n)} = g_{k-1,i}^{(n)} + g_{k-1,k}^{(n)} \frac{g_{k-1,i}^{(n)} - g_{k-1,i}^{(n+1)}}{g_{k-1,k}^{(n+1)} - g_{k-1,k}^{(n)}}, \qquad i = k+1, k+2, \ldots. \tag{2.2}$$

It can very easily be proved, by using Sylvester's determinantal identity [8, Vol. 1, p. 32] that

$$E_k^{(n)} = E_k(S_n).$$

Usually the quantities $E_k^{(n)}$ are displayed in a two-dimensional array, called the E-array, as follows:

$$\begin{array}{llll} E_0^{(0)} = S_0 & & & \\ E_0^{(1)} = S_1 & E_1^{(0)} & & \\ E_0^{(2)} = S_2 & E_1^{(1)} & E_2^{(0)} & \\ E_0^{(3)} = S_3 & E_1^{(2)} & E_2^{(1)} & E_3^{(0)} \\ \vdots & \vdots & \vdots & \ddots \end{array}$$

The $g_{k,i}^{(n)}$'s can also be placed in similar arrays, the g_i-arrays, for a fixed value of the index i, where k indicates the column starting from zero. It is easy to check that $g_{k,i}^{(n)} = 0$ for $k \geq i$, and thus the g_i-array only contains the columns 0 to $i - 1$.

There are two methods for using, and thus for programming, convergence acceleration methods. The first one is to use them a posteriori, that is, to compute a fixed number of terms of the sequence (S_n) and then to apply the algorithm by computing each column of the array from the preceding one. This use is very easy to program, but it has the main disadvantage that all the computations must be started again from the beginning if one wants to add new terms to the initial sequence (S_n).

The second use, which is much more convenient in practice, is to use the algorithm "in parallel" with the computation of the successive terms of the original sequence (S_n): After the computation of each new term of the sequence

(S_n), one has to build the arrays as far as possible by computing each row from the preceding one and then to go back to the computation of the next term of (S_n). Of course, this way of implementing the algorithms is much more complicated to program, but is it the only reasonable way. For a complete description of this use see [1] and for its special application to the E-algorithm see [3].

Let us now describe the arguments of the subroutine EXTRAP for the parallel use of the E-algorithm.

The subroutine must be included in a loop where the successive terms of the sequence (S_n) are computed. Before the beginning of the loop the integer parameter INIT must be set to zero. Then the value of INIT is modified by the subroutine, and thus INIT might be set again to zero if one wants to use the subroutine later for another sequence.

In the subroutine, $E(I)$ contains $E_{I-1}^{(K-I)}$ for $I = 1, \ldots, \min(K, \text{MAXCOL} + 1)$ after the Kth call, while $G(I, J)(I = 1, \ldots, \min(K, \text{MAXCOL}))$ contains $g_{J-1,I}^{(K-J)}$ for $J = 1, \ldots, I - 1$, and $g_{I-1,J}^{(J-I)}$ for $J = I, \ldots, \min(K, \text{MAXCOL} + 1)$.

A partial listing of the subroutine EXTRAP follows:

```
      SUBROUTINE EXTRAP(INIT, EPS, MAXCOL, SK, GINIT, RESULT, INFO)     EXT   1Ø
C                                                                        EXT   2Ø
C   STATEMENT OF PURPOSE                                                 EXT   3Ø
C   ********************                                                 EXT   4Ø
C                                                                        EXT   5Ø
C   THE SUBPROGRAM EXTRAP() IS A SUBROUTINE TO IMPLEMENT THE E-ALGORITHMEXT   6Ø
C   FOR SEQUENCE EXTRAPOLATION AND THE MUHLBACH-NEVILLE-AITKEN ALGORITHMEXT   7Ø
C   FOR THE GENERAL INTERPOLATION PROBLEM.                               EXT   8Ø
C                                                                        EXT   9Ø
C                                                                        EXT  1ØØ
C   DESCRIPTION OF THE PARAMETERS OF THE SUBROUTINE EXTRAP()             EXT  11Ø
C   ********************************************************           EXT  12Ø
C                                                                        EXT  13Ø
C                                                                        EXT  14Ø
C INIT   INTEGER PARAMETER TO BE SET TO ZERO BEFORE THE FIRST            EXT  15Ø
C        CALL OF THE SUBROUTINE.  DUE TO THE FACT THAT AFTER THE         EXT  16Ø
C        COMPUTATION OF EACH ROW OF THE E-ARRAY ONE HAS TO EXIT THE      EXT  17Ø
C        SUBROUTINE TO COMPUTE THE NEW VALUES OF THE DATA SK AND         EXT  18Ø
C        GINIT(*), ALL THE VARIABLES INTERNAL TO THE SUBROUTINE MUST     EXT  19Ø
C        REMAIN INTACT BETWEEN THE CALLS. THE SAME IS TRUE FOR THE       EXT  2ØØ
C        ARGUMENTS INIT , MAXCOL AND INFO.  THE VALUE OF EPS CAN BE      EXT  21Ø
C        CHANGED.  DURING THE FIRST CALL OF THE SUBROUTINE INIT IS       EXT  22Ø
C        CHANGED TO 1. TO USE THE SUBROUTINE FOR A NEW APPLICATION OF    EXT  23Ø
C        THE ALGORITHM (AND NOT FOR THE SUBSEQUENT CALLS CORRESPONDING   EXT  24Ø
C        TO THE SAME APPLICATION) INIT WILL HAVE TO BE SET AGAIN TO Ø.   EXT  25Ø
C                                                                        EXT  26Ø
C EPS    THE AIM OF THIS PARAMETER IS TO AVOID A DIVISION BY ZERO.       EXT  27Ø
C        WHEN THE ABSOLUTE VALUE OF A DENOMINATOR IS LESS THAN EPS THE   EXT  28Ø
C        PARAMETER INFO IS SET TO 1.                                     EXT  29Ø
C                                                                        EXT  3ØØ
C MAXCOL INTEGER PARAMETER GIVING THE INDEX OF THE LAST COLUMN           EXT  31Ø
C        OF THE E-ARRAY THAT THE USER WANTS TO COMPUTE (SEE              EXT  32Ø
C        BELOW IN THE EXPLANATIONS FOR THE PARAMETER RESULT.)            EXT  33Ø
C                                                                        EXT  34Ø
C SK     DOUBLE PRECISION ARGUMENT WHICH MUST CONTAIN THE VALUE          EXT  35Ø
C        OF S    BEFORE THE K-TH CALL OF THE SUBROUTINE.                 EXT  36Ø
C            K-1                                                         EXT  37Ø
C                                                                        EXT  38Ø
C GINIT  DOUBLE PRECISION ARRAY WHICH MUST CONTAIN THE VALUES OF         EXT  39Ø
C        G (Ø) BEFORE THE FIRST CALL OF THE SUBROUTINE, G (1)            EXT  4ØØ
C         1                                                 1            EXT  41Ø
C        BEFORE THE SECOND CALL, G (K-1), G (K-1),...., G   (K-1),       EXT  42Ø
C                                 1        2             K-2            EXT  43Ø
C        G   (Ø), G   (1),...., G   (K-1) BEFORE THE K-TH CALL           EXT  44Ø
C         K-1      K-1           K-1                                     EXT  45Ø
C        IF MAXCOL+1 J= K J= 3 AND G (K-1),...., G      (K-1) IF         EXT  46Ø
C                                   1            MAXCOL                  EXT  47Ø
C        K J MAXCOL + 1                                                  EXT  48Ø
C.                                                                       EXT  49Ø
C RESULT  DOUBLE PRECISION ARGUMENT CONTAINING THE ANSWER GIVEN BY       EXT  5ØØ
C        THE SUBROUTINE AFTER THE CALL. RESULT SUCCESSIVELY CONTAINS     EXT  51Ø
C         (Ø)     (Ø)          (Ø)        (1)        (2)                 EXT  52Ø
C        E    , E    ,...., E       , E        , E                      EXT  53Ø
C         Ø      1           MAXCOL     MAXCOL     MAXCOL               EXT  54Ø
C                                                                        EXT  55Ø
C INFO   INTEGER PARAMETER WHICH, ON NORMAL EXIT, HAS THE VALUE Ø.       EXT  56Ø
C        A NEGATIVE VALUE SIGNALS THAT DIMENSIONS ARE TOO SMALL IN       EXT  57Ø
C        THE SUBROUTINE. THE VALUE 1 INDICATES A DIVISION BY A           EXT  58Ø
C        QUANTITY SMALLER THAN EPS IN ABSOLUTE VALUE. IF INFO HAS A      EXT  59Ø
C        VALUE DIFFERENT FROM ZERO IT IS IMPOSSIBLE TO ENTER AGAIN       EXT  6ØØ
C        INTO EXTRAP(). TO USE IT AGAIN THE PARAMETER INIT MUST BE       EXT  61Ø
C        SET TO Ø.                                                       EXT  62Ø
```

Let us give an example of the use of this subroutine. It corresponds to the choice $g_i(n) = \Delta S_{n+i-1}$ (Shanks' transformation) for the sequence $S_n = 1/(n+1)$. For this sequence it is known that $E_k^{(0)} = \epsilon_{2k}^{(0)} = 1/(k+1)^2$.

```
C     PROGRAM TESTD (OUTPUT,TAPE6=OUTPUT)                              MAN   1Ø
C     TEST DRIVER FOR THE EXTRAP SUBROUTINE                            MAN   2Ø
C     IN THE E-ALGORITHM                                               MAN   3Ø
C                                                                      MAN   4Ø
      INTEGER I, MAXCOL, INIT, INFO, K, M, MPI, NI                     MAN   5Ø
      DOUBLE PRECISION KPI, SK, RESULT, S, GINIT, EPS, T               MAN   6Ø
      DIMENSION GINIT(5Ø)                                              MAN   7Ø
C                                                                      MAN   8Ø
      DATA NOUT /6/, INIT /Ø/, MAXCOL /5/, EPS /1.D-3Ø/                MAN   9Ø
C                                                                      MAN  1ØØ
      DO 4Ø K=1,2Ø                                                     MAN  11Ø
C                                                                      MAN  12Ø
C     COMPUTATION OF S                                                 MAN  13Ø
C                     K-1                                              MAN  14Ø
C                                                                      MAN  15Ø
        T = K                                                          MAN  16Ø
        SK = 1.DØ/T                                                    MAN  17Ø
C                                                                      MAN  18Ø
C     COMPUTATION OF G (K-1)                                           MAN  19Ø
C                     I                                                MAN  2ØØ
C                                                                      MAN  21Ø
        M = MAXØ(1,K-2)                                                MAN  22Ø
        DO 1Ø I=1,M                                                    MAN  23Ø
          KPI = K + I                                                  MAN  24Ø
          GINIT(I) = -1.DØ/(KPI-1.DØ)/KPI                              MAN  25Ø
   1Ø   CONTINUE                                                       MAN  26Ø
        IF (K.LT.3 .OR. K.GT.MAXCOL+1) GO TO 3Ø                        MAN  27Ø
C                                                                      MAN  28Ø
C     COMPUTATION OF G   (I-1)                                         MAN  29Ø
C                     K-1                                              MAN  3ØØ
C                                                                      MAN  31Ø
        DO 2Ø I=1,K                                                    MAN  32Ø
          KPI = K + I                                                  MAN  33Ø
          MPI = M + I                                                  MAN  34Ø
          GINIT(MPI) = -1.DØ/(KPI-2.DØ)/(KPI-1.DØ)                    MAN  35Ø
   2Ø   CONTINUE                                                       MAN  36Ø
   3Ø   CALL EXTRAP(INIT, EPS, MAXCOL, SK, GINIT, RESULT, INFO)       MAN  37Ø
        IF (INFO.LT.Ø) WRITE (NOUT,99998)                              MAN  38Ø
        IF (INFO.EQ.1) WRITE (NOUT,99997)                              MAN  39Ø
        T = K                                                          MAN  4ØØ
        S = 1.DØ/T**2                                                  MAN  41Ø
        T = (MAXCOL+1)*K                                               MAN  42Ø
        IF (K.GT.MAXCOL+1) S = 1.DØ/T                                  MAN  43Ø
        WRITE (NOUT,99999) RESULT, S                                   MAN  44Ø
   4Ø CONTINUE                                                         MAN  45Ø
      STOP                                                             MAN  46Ø
99999 FORMAT (3D26.12)                                                 MAN  47Ø
99998 FORMAT (46H DIMENSIONS TOO SMALL IN THE SUBROUTINE EXTRAP)       MAN  48Ø
99997 FORMAT (42H DIVISION BY ZERO IN THE SUBROUTINE EXTRAP)           MAN  49Ø
      END                                                              MAN  5ØØ
```

This program gives the results shown in Table I on a computer with precision 10^{-18} (DEC 10). The numbers printed are $E_0^{(0)}, \ldots, E_5^{(0)}, E_5^{(1)}, \ldots, E_5^{(14)}$, where the exact values are given by $E_k^{(n)} = 1/((n+k+1)(k+1))$. The E-algorithm can be quite sensitive to the propagation of rounding errors. Several possibilities exist for their control. The first one is to compute the "amplification factors" defined by Håvie [9] which can be used to obtain an upper bound for the amplification of the initial errors and which can be considered as a measurement of the numerical stability of the algorithm. The second possibility consists in using the particular rules given in [4] although they are quite difficult to program and will lead to a much longer program. The third possibility is to use an exact arithmetic such as p-adic or rational arithmetic. Such an arithmetic is actually under consideration.

A quite important practical problem when using convergence acceleration methods is that of the termination criteria. For some special sequences and some particular methods such a criterion can be obtained (see, for example, the discussion on totally monotonic and totally oscillating sequences and their treatment by Shanks's transformation in [1]). In the general case some theorems exist [3, Theorem 3], but in practice the computations are stopped when two successive elements of the same column or the same diagonal of the E-array are nearly equal.

To end this section let us mention that the relationships of the E-algorithm, as they are written in [3], are numerically unstable. These relations must be written

Table I

Computed values	Exact values
0.100000000000D+01	0.100000000000D+01
0.250000000000D+00	0.250000000000D+00
0.111111111111D+00	0.111111111111D+00
0.625000000000D−01	0.625000000000D−01
0.400000000000D−01	0.400000000000D−01
0.277777777778D−01	0.277777777778D−01
0.238095238096D−01	0.238095238095D−01
0.208333333332D−01	0.208333333333D−01
0.185185185190D−01	0.185185185185D−01
0.166666666650D−01	0.166666666667D−01
0.151515151566D−01	0.151515151515D−01
0.138888888769D−01	0.138888888889D−01
0.128205128394D−01	0.128205128205D−01
0.119047618869D−01	0.119047619048D−01
0.111111111178D−01	0.111111111111D−01
0.104166666706D−01	0.104166666667D−01
0.980392156462D−02	0.980392156863D−02
0.925925926345D−02	0.925925925926D−02
0.877192979360D−02	0.877192982456D−02
0.833333342965D−02	0.833333333333D−02

as they are in (2.1) and (2.2) and in the subroutine. In that way they will be much more stable. For example, if the unstable version of these rules is used, then, for the preceding example, we shall obtain for the last entry a result that is less accurate by three digits.

3. THE MÜHLBACH–NEVILLE-AITKEN ALGORITHM

Let $\{f_i\}_{i\geq 0}$ be a family of linearly independent elements of a real vector space E. $\{f_i\}$ is assumed to be a complete Chebyshev system with respect to the linear functionals $\{L_j\}$, which means that all the Gram determinants $|L_j(f_i)|_{i,j=0,\ldots,k}$ are not zero whenever $L_0, \ldots, L_k$ are linearly independent.

The problem is to construct a generalized polynomial

$$P_k = a_0 f_0 + \cdots + a_k f_k$$

such that

$$L_i(P_k) = w_i, \qquad i = 0, \ldots, k,$$

where the w_i's are given numbers not all zero.

More generally one wants to construct

$$P_k^{(n)} = a_0 f_0 + \cdots + a_k f_k$$

such that

$$L_i(P_k^{(n)}) = w_i, \qquad i = n, \ldots, n + k.$$

This is the general interpolation problem as described, for example, by Davis [7].

Usually on a computer (and also often in numerical analysis) one deals with numbers and not with elements of a vector space. This means that, in practice, one has to compute the numerical values $L(P_k^{(n)})$ where L is a linear functional on E. (If E is a vector space of functions, then, for example, $L(f)$ is the value of the function f at some prescribed point x.)

Mühlbach [12, 13] showed that these values can be recursively computed by an algorithm that generalizes the Neville-Aitken scheme. For this reason it has been called the Mühlbach–Neville-Aitken algorithm (briefly, the MNA-algorithm). A very simple proof of the MNA-algorithm has been given in [4] using Sylvester's determinantal identify. Except for the initializations, it is similar to the E-algorithm.

$$L(P_0^{(n)}) = w_n \frac{L(f_0)}{L_n(f_0)}$$

$$L(f_{0,i}^{(n)}) = L_n(f_i)\frac{L(f_0)}{L_n(f_0)} - L(f_i)$$

$$L(P_k^{(n)}) = \frac{L(f_{k-1,k}^{(n+1)})L(P_{k-1}^{(n)}) - L(f_{k-1,k}^{(n)})L(P_{k-1}^{(n+1)})}{L(f_{k-1,k}^{(n+1)}) - L(f_{k-1,k}^{(n)})}$$

$$L(f_{k,i}^{(n)}) = \frac{L(f_{k-1,k}^{(n+1)})L(f_{k-1,i}^{(n)}) - L(f_{k-1,k}^{(n)})L(f_{k-1,i}^{(n+1)})}{L(f_{k-1,k}^{(n+1)}) - L(f_{k-1,k}^{(n)})}.$$

Thus the subroutine EXTRAP can still be used for computing the values of $L(P_k^{(n)})$. For example, if we choose

$$f_0(x) = e^{-x} \qquad f_1(x) = \sin x \qquad f_2(x) = x^2 \qquad f_3(x) = \log x$$

$$L_0(f_i) = f_i(1) \qquad L_1(f_i) = f_i'(2) \qquad L_2(f_i) = \int_0^1 f_i(x)\,dx \qquad L_3(f_i) = f_i''(1)$$

$$w_i = L_i(f) \quad i = 0, \ldots, 3 \quad \text{with} \quad f(x) = 3f_0(x) + 4f_1(x) - f_2(x) + f_3(x)$$

and $L(P_k^{(n)}) = P_k^{(n)}(x)$, then the calling sequence is as follows:

```
C     PROGRAM TESTD (OUTPUT,TAPE6=OUTPUT)                                    MAN   1Ø
C     TEST DRIVER FOR THE EXTRAP SUBROUTINE IN THE                           MAN   2Ø
C     MUHLBACH-NEVILLE-AITKEN ALGORITHM                                      MAN   3Ø
C                                                                            MAN   4Ø
      INTEGER I, INIT, INFO, K, L, M, MPI, N, NI                             MAN   5Ø
      DOUBLE PRECISION A, GINIT, F, P, RESULT, SK, W, X, EPS, T              MAN   6Ø
      DIMENSION A(1Ø,1Ø), GINIT(4Ø), F(1Ø), W(1Ø)                            MAN   7Ø
C                                                                            MAN   8Ø
      DATA N /4/, NOUT /6/, EPS /1.D-3Ø/                                     MAN   9Ø
C                                                                            MAN  1ØØ
C     N IS THE NUMBER OF TERMS IN THE LINEAR COMBINATION                     MAN  11Ø
C                                                                            MAN  12Ø
C     W(I) MUST CONTAIN W                                                    MAN  13Ø
C                       I-1                                                  MAN  14Ø
C                                                                            MAN  15Ø
      W(1) = 3.DØ*DEXP(-1.DØ) + 4.DØ*DSIN(1.DØ) - 1.DØ                       MAN  16Ø
      W(2) = -3.DØ*DEXP(-2.DØ) + 4.DØ*DCOS(2.DØ) - 3.5DØ                     MAN  17Ø
      W(3) = 6.DØ - 1.DØ/3.DØ - 3.DØ*DEXP(-1.DØ) - 4.DØ*DCOS(1.DØ)          MAN  18Ø
      W(4) = 3.DØ*DEXP(-1.DØ) - 4.DØ*DSIN(1.DØ) - 3.DØ                       MAN  19Ø
C                                                                            MAN  2ØØ
C     A(I,J) MUST CONTAIN L   (F    )                                        MAN  21Ø
C                          I-1  J-1                                          MAN  22Ø
C                                                                            MAN  23Ø
      A(1,1) = DEXP(-1.DØ)                                                   MAN  24Ø
      A(2,1) = -DEXP(-2.DØ)                                                  MAN  25Ø
      A(3,1) = 1.DØ - A(1,1)                                                 MAN  26Ø
      A(4,1) = A(1,1)                                                        MAN  27Ø
      A(1,2) = DSIN(1.DØ)                                                    MAN  28Ø
      A(2,2) = DCOS(2.DØ)                                                    MAN  29Ø
      A(3,2) = 1.DØ - DCOS(1.DØ)                                             MAN  3ØØ
      A(4,2) = -A(1,2)                                                       MAN  31Ø
      A(1,3) = 1.DØ                                                          MAN  32Ø
      A(2,3) = 4.DØ                                                          MAN  33Ø
      A(3,3) = 1.DØ/3.DØ                                                     MAN  34Ø
      A(4,3) = 2.DØ                                                          MAN  35Ø
      A(1,4) = Ø.DØ                                                          MAN  36Ø
      A(2,4) = Ø.5DØ                                                         MAN  37Ø
      A(3,4) = -1.DØ                                                         MAN  38Ø
      A(4,4) = -1.DØ                                                         MAN  39Ø
C                                                                            MAN  4ØØ
      DO 5Ø L=1,1Ø                                                           MAN  41Ø
        T = L                                                                MAN  42Ø
        X = T*Ø.2DØ                                                          MAN  43Ø
C                                                                            MAN  44Ø
C     F(I) MUST CONTAIN F                                                    MAN  45Ø
C                       I-1                                                  MAN  46Ø
C                                                                            MAN  47Ø
        F(1) = DEXP(-X)                                                      MAN  48Ø
        F(2) = DSIN(X)                                                       MAN  49Ø
        F(3) = X*X                                                           MAN  5ØØ
        F(4) = DLOG(X)                                                       MAN  51Ø
        INIT = Ø                                                             MAN  52Ø
        DO 4Ø K=1,N                                                          MAN  53Ø
          SK = W(K)*F(1)/A(K,1)                                              MAN  54Ø
          M = MAXØ(1,K-2)                                                    MAN  55Ø
C                                                                            MAN  56Ø
```

```
C                    (K-1)                                                  MAN  57Ø
C     COMPUTATION OF F                                                      MAN  58Ø
C                    Ø,I                                                    MAN  59Ø
C                                                                           MAN  6ØØ
          DO 1Ø I=1,M                                                       MAN  61Ø
            GINIT(I) = A(K,I+1)*F(1)/A(K,1) - F(I+1)                        MAN  62Ø
   1Ø     CONTINUE                                                          MAN  63Ø
          IF (K.LT.3) GO TO 3Ø                                              MAN  64Ø
C                                                                           MAN  65Ø
C                    (I-1)                                                  MAN  66Ø
C     COMPUTATION OF F                                                      MAN  67Ø
C                    Ø,K-1                                                  MAN  68Ø
C                                                                           MAN  69Ø
          DO 2Ø I=1,K                                                       MAN  7ØØ
            MPI = M + I                                                     MAN  71Ø
            GINIT(MPI) = A(I,K)*F(1)/A(I,1) - F(K)                          MAN  72Ø
   2Ø     CONTINUE                                                          MAN  73Ø
   3Ø     CALL EXTRAP(INIT, EPS, N-1, SK, GINIT, RESULT, INFO)              MAN  74Ø
   4Ø   CONTINUE                                                           MAN  75Ø
        IF (INFO.LT.Ø) WRITE (NOUT,99998)                                   MAN  76Ø
        IF (INFO.EQ.1) WRITE (NOUT,99997)                                   MAN  77Ø
        P = 3.DØ*F(1) + 4.DØ*F(2) - F(3) + F(4)                             MAN  78Ø
        WRITE (NOUT,99999) X, P, RESULT                                     MAN  79Ø
   5Ø CONTINUE                                                              MAN  8ØØ
      STOP                                                                  MAN  81Ø
99999 FORMAT (3D26.12)                                                      MAN  82Ø
99998 FORMAT (46H DIMENSIONS TOO SMALL IN THE SUBROUTINE EXTRAP)            MAN  83Ø
99997 FORMAT (42H DIVISION BY ZERO IN THE SUBROUTINE EXTRAP)                MAN  84Ø
      END                                                                   MAN  85Ø
```

Table II

x	$L(f)$	$L(P_4^{(0)})$
0.20000000000000D+00	0.16014316699801D+01	0.16014316699801D+01
0.40000000000000D+00	0.24923427754674D+01	0.24923427754674D+01
0.60000000000000D+00	0.30341791780962D+01	0.30341791780962D+01
0.80000000000000D+00	0.33542677046355D+01	0.33542677046355D+01
0.10000000000000D+01	0.34695222627459D+01	0.34695222627459D+01
0.12000000000000D+01	0.33740605363995D+01	0.33740605363995D+01
0.14000000000000D+01	0.30580620483999D+01	0.30580620483999D+01
0.16000000000000D+01	0.25139875953957D+01	0.25139875953957D+01
0.18000000000000D+01	0.17390738530797D+01	0.17390738530797D+01
0.20000000000000D+01	0.73634273757251D+00	0.73634273757251D+00

Of course, $L(P_4^{(0)}) = L(f)$ for all x, as can be checked from the numerical results shown in Table II.

The MNA-algorithm can also be used to implement another convergence acceleration process due to Germain-Bonne [9, pp. 33–43] that is based on generalized inverse interpolation.

4. LEAST SQUARES APPROXIMATION AND EXTRAPOLATION

Let E be an inner product space. The least squares approximation of $f \in E$ by a combination of the given independent elements $f_0, \ldots, f_k$ of E consists in determining $a_0, \ldots, a_k$ such that

$$\| f - (a_0 f_0 + \cdots + a_k f_k) \|$$

be minimum where $\| \cdot \|^2 = (\cdot, \cdot)$. It is well known (e.g., see [7, p. 182]) that the solution $F_k = a_0 f_0 + \cdots + a_k f_k$ of this minimization problem is given by a ratio of determinants completely similar to the one defining $P_k^{(0)}$ in the general interpolation problem; one only has to replace $L_i(f_j)$ by (f_i, f_j) and w_i by (f_i, f). If L is a linear functional on E, then the sequence $(L(F_k))$ can be obtained via the MNA algorithm with the initializations

$$P_0^{(n)} = (f_n, f) \frac{L(f_0)}{(f_n, f_0)}$$

$$f_{0,i}^{(n)} = (f_n, f_i) \frac{L(f_0)}{(f_n, f_0)} - L(f_i).$$

We shall obtain

$$P_k^{(0)} = L(F_k).$$

A special case of this general problem is to find F_k, which satisfies the system of linear equations

$$L_i(F_k) = w_i, \qquad i = 0, \ldots, m \geq k$$

in the least squares sense. The solution of this problem can be obtained by the MNA-algorithm where the scalar product is defined by

$$(f, g) = \sum_{i=0}^{m} L_i(f) L_i(g).$$

Another special case is the biconjugate gradient method for solving systems of linear equations whose successive iterates are still given by a similar ratio of determinants [2, p. 87] and which can therefore be computed by the MNA-algorithm.

The subroutine EXTRAP can be used for implementing all these least squares approximation problems. This subroutine can also be used for a convergence acceleration process due to Cordellier [6] that is based on least squares. Let us explain this process.

The E-transformation of sequences described in Section 2 was obtained by assuming that the given sequence (S_n) satisfied

$$S_{n+i} = S + a_1 g_1(n + i) + \cdots + a_k g_k(n + i)$$

for $i = 0, \ldots, k$ and then solving this system for the unknown S. Cordellier's idea consists of writing this system for $i = 0, \ldots, m \geq k$ and then solving it in the least squares sense for the unknown S. This least squares solution can be obtained by the E-algorithm as we now see. We set

$$g_i = (g_i(n); \ldots; g_i(n + m))^{\mathrm{T}}, \qquad i \geq 1$$

$$g_0 = (1; \ldots; 1)^{\mathrm{T}}$$

$$V = (S_n; \ldots; S_{n+m})^{\mathrm{T}}.$$

Thus the system becomes

$$\begin{pmatrix} 1 & \frac{(g_0, g_1)}{(g_0, g_0)} & \cdots & \frac{(g_0, g_k)}{(g_0, g_0)} \\ - & - & - & - \\ 1 & \frac{(g_k, g_1)}{(g_k, g_0)} & \cdots & \frac{(g_k, g_k)}{(g_k, g_0)} \end{pmatrix} \begin{pmatrix} S \\ a_1 \\ \vdots \\ a_k \end{pmatrix} = \begin{pmatrix} \frac{(g_0, V)}{(g_0, g_0)} \\ \vdots \\ \frac{(g_k, V)}{(g_k, g_0)} \end{pmatrix}$$

The value of S thus obtained will depend on m, k, and n, and we denote it by ${}_mE_k^{(n)}$. Since the preceding system has the same form as the system implicitly solved by the E-algorithm, we can use it to obtain the sequence ${}_mE_k^{(n)}$ for $k = 0, \ldots, m$ and n and m fixed. In this case the initializations must be

$$E_0^{(n)} = \frac{(g_n, V)}{(g_n, g_0)}$$

$$g_{0,i}^{(n)} = \frac{(g_n, g_i)}{(g_n, g_0)}, \qquad i \geq 1$$

and we shall get

$$E_k^{(0)} = {}_mE_k^{(n)}.$$

If the value of n or m is modified, then the definition of the scalar product is also modified and all the computations have to be started again from the beginning. For $k = m$ the least squares extrapolated value will be identical to the value obtained by the E-transformation as described in Section 2. This is also

Table III

k	${}_4E_k^{(0)}$ (Perturbed sequence and perturbed g_i)	R.E.	${}_4E_k^{(0)}$ (Perturbed sequence and unperturbed g_i)	R.E.	${}_4E_k^{(0)}$ (Unperturbed sequence)
0	0.4567459	1.7	0.4567459	1.7	0.4566666
1	0.1782071	2.3	0.1782336	3.8	0.1781660
2	0.0937578	6.4	0.0938553	4.0	0.0938178
3	0.0581327	32.4	0.0583042	3.0	0.0583216
4	0.0398319	42.0	0.0398976	25.6	0.0399999

true for the value obtained by least squares approximation that will be equal to the value obtained by interpolation when $m = k$.

The basic advantage of least squares extrapolation is to be less sensitive to small perturbations in the sequence (S_n) to be accelerated.

Let us take the same example as in Section 2 and let us add a perturbation of magnitude less than 10^{-4} to the sequence (S_n).

Since the sequences (g_i) depend on (S_n), a perturbation of (S_n) will also modify the (g_i). Thus two different experiments have been conducted: with the (g_i) automatically modified by the perturbation of (S_n) and with unmodified (g_i). Columns R.E. in Table III contain the values of the relative errors with respect to the unperturbed case multiplied by 10000. For $n = 0$ and $m = 4$ we get the results also shown in the table. As we can verify, the ${}_4E_k^{(0)}$'s $(k < 4)$ are less sensitive to the perturbation than ${}_4E_4^{(0)} = E_4^{(0)} = 0.04$. It must be noticed that the results obtained are very sensitive to the precision on the computation of the scalar products appearing in the initializations.

ACKNOWLEDGMENTS

I am very much indebted to the two referees who reviewed this paper. The first one provided me with the numerical example showing the influence of the computer's precision, while the second gave me many valuable suggestions for improving the presentation of the paper. I also want to convey my gratitude to Prof. Å. Björck, who suggested many improvements and who, with the help of one of his assistants, modified my programs and checked them to be portable. Thanks are also due to Dr. R.J. Hanson and Prof. J.R. Rice for their numerous comments that greatly helped me to polish the paper and enhance the quality of the subroutine.

REFERENCES

1. BREZINSKI, C. *Algorithmes d'accélération de la convergence. Etude Numérique.* Editions Technip, Paris, 1978.
2. BREZINSKI, C. Padé-type approximation and general orthogonal polynomials. *International Series of Numerical Mathematics*, Vol. 50, Birkhaüser-Verlag, Basel, Switzerland, 1980.
3. BREZINSKI, C. A general extrapolation algorithm. *Numer. Math. 35* (1980), 175–187.
4. BREZINSKI, C. The Mühlbach-Neville-Aitken algorithm and some extensions. *BIT 20* (1980), 444–451.
5. CORDELLIER, F. Analyse numérique des transformations de suites et de séries. Thèse, Université de Lille, Lille, France, to be published.
6. CORDELLIER, F. Une méthode d'accélération de la convergence basée sur l'approximation au sens des moindres carrés. Séminaire d'analyse numérique et d'optimisation (Université de Lille, Lille, France, March 13, 1975).
7. DAVIS, P.J. *Interpolation and Approximation.* Blaisdell, New York, 1961.
8. GANTMACHER, F.R. *The Theory of Matrices.* Chelsea, New York, 1959.
9. GERMAIN-BONNE, B. Estimation de la limite de suites et formalisation de procédés d'accélération de convergence. Thèse, Université de Lille, Lille, France, 1978.
10. HÅVIE, T. Generalized Neville type extrapolation schemes. *BIT 19* (1979), 204–213.
11. LEVIN, D. Development of non-linear transformations for improving convergence of sequences. *Int. J. Comput. Math. B3* (1973), 371–388.
12. MÜHLBACH, G. Neville-Aitken algorithms for interpolation by functions of Chebyshev-systems in the sense of Newton and in a generalized sense of Hermite. In *Theory of Approximation with Applications*, A.G. Law and B.N. Sahney (Eds.), Academic, New York, 1976.

13. MÜHLBACH, G. The general Neville-Aitken algorithm and some applications. *Numer. Math. 31* (1978), 97–110.
14. SHANKS, D. Non linear transformations of divergent and slowly convergent sequences. *J. Math. Phys. 34* (1955), 1–42.
15. WIMP, J. *Sequence Transformations and Their Applications.* Academic, New York, 1981.
16. WYNN, P. On a device for computing the $e_m(S_n)$ transformation. *Math. Tables and Other Aids to Computations 10* (1956), 91–96.

ALGORITHM

[Part of the listing is printed above. The complete listing is available from the ACM Distribution Service .]

586-P 1- 0

ALGORITHM 586
ITPACK 2C: A FORTRAN Package for Solving Large Sparse Linear Systems by Adaptive Accelerated Iterative Methods

DAVID R. KINCAID, JOHN R. RESPESS, and DAVID M. YOUNG
University of Texas at Austin
and
ROGER G. GRIMES
Boeing Computer Services Company

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*linear systems (direct and iterative methods); sparse and very large systems*; G.4 [**Mathematics of Computing**]: Mathematical Software—*algorithm analysis; certification and testing*; G.m [**Mathematics of Computing**]: Miscellaneous—*FORTRAN*
General Terms: Algorithms, Documentation
Additional Key Words and Phrases: Iterative methods, numerical software, sparse matrix

1. INTRODUCTION

ITPACK 2C is a collection of seven FORTRAN subroutines for solving large sparse linear systems by adaptive accelerated iterative algorithms. Basic iterative procedures, such as the Jacobi method, the successive overrelaxation method, the symmetric successive overrelaxation method, and the RS method for the reduced system are combined, where possible, with acceleration procedures, such as Chebyshev (semi-iteration) and conjugate gradient, for rapid convergence. Automatic selection of the acceleration parameters and the use of accurate stopping criteria are major features of this software package. While the ITPACK routines can be called with any linear system containing positive diagonal elements, they are the most successful in solving systems with symmetric positive definite or mildly nonsymmetric coefficient matrices.

For several years, we have been involved with the development and use of research-oriented programs using iterative algorithms for solving large sparse

$$Au = b$$

with positive diagonal elements. One solves for the N component unknown vector u given the $N \times N$ nonsingular coefficient matrix A and the N component right-hand side vector b. The current ITPACK software package of subroutines, version 2C, provides for the use of seven alternative iterative procedures. While these subroutines are not designed as production software, they should successfully handle industrial problems of moderate size, that is, ones that fit in high-speed

Received 22 May 1979; revised 15 May 1982; accepted 16 May 1982
Work on this project was supported in part by National Science Foundation Grant MCS 79-19829 at The University of Texas at Austin.
Authors' addresses: D. R. Kincaid, J. R. Respess, and D. M. Young, Center for Numerical Analysis & Computation Center, RLM Bldg. 13.150, University of Texas, Austin, TX 78712; R. G. Grimes, Boeing Computer Services Co., 565 Andover Park West, Tukwila, WA 98188.

memory. This package is written in 1966 American National Standard FORTRAN code. It has been tested over a wide variety of computer systems using various FORTRAN compilers, including one which is FORTRAN-77 compatible (see Acknowledgments).

The seven iterative solution modules are based on several basic iterative procedures, such as the Jacobi method, the successive overrelaxation (SOR) method, the symmetric SOR (SSOR) method, and the RS method for the reduced system. With the exception of SOR, the convergence of these basic methods are accelerated by Chebyshev (semi-iteration, SI) or conjugate gradient (CG) acceleration. All methods are available with adaptive parameter estimation and automatic stopping tests. When using the RS method it is required that the linear system be reordered into a "red-black" system [6, 13].[1] A switch to compute, if possible, the red-black indexing, permute the linear system, and permute associated vectors is provided.

The successful convergence of iterative methods may be dependent on conditions that are difficult to determine in advance. For example, determining whether the coefficient matrix is positive definite can be as costly to check as solving the system. On the other hand, some conditions affecting convergence, such as positive diagonal elements, diagonal dominance, and symmetry are relatively easy to verify. For some applications, the theory to guarantee the convergence of an iterative method may not exist. The algorithms in ITPACK have been tested most extensively for linear systems arising from elliptic partial differential equations. The routines can be applied formally to any linear system which fits in high-speed memory. However, rapid convergence, and indeed convergence itself cannot be guaranteed unless the matrix of the system is symmetric and positive definite. Success can be expected, though not guaranteed, for mildly nonsymmetric systems. In other words, iterative methods may not converge when applied to systems with coefficient matrices which are completely general with no special properties.

This article discusses the usage of ITPACK and gives a few test results. The description of the iterative methods is given in [4]. The underlying theory on which the iterative algorithms are based is described in [6]. A survey of the iterative methods in ITPACK is presented in [12].

Throughout this paper, we adopt notation such as **SOR( )** when referring to a subroutine and **A(*)** for a single-dimensioned array. The residual vector is $b - Au$ for the linear system $Au = b$ and the pseudo-residual vector is $Gu^{(n)} + k - u^{(n)}$ for a basic iterative method of the form $u^{(n+1)} = Gu^{(n)} + k$. The smallest and largest eigenvalues of the iteration matrix G are denoted $m(G)$ and $M(G)$, respectively.

2. SPARSE MATRIX STORAGE

The sparse storage scheme used in ITPACK is a common one. It is a row-wise representation of the nonzero entries in the coefficient matrix of the linear system. For a nonsymmetric coefficient matrix, all of the nonzero values in each row are stored in a contiguous block of data in a real-valued array **A(*)**. If the matrix is symmetric, computer memory can be saved by storing only the nonzero entries in each row on and above the main diagonal. For either nonsymmetric or symmetric sparse storage, associated column numbers are stored in an integer-valued array **JA(*)** such that **JA(K)** is the column number for the value **A(K)**.

[1] In this ordering, the components of the unknown vector u are considered as either "red" or "black". A "red-black ordering" is any ordering such that every black unknown follows all of the red unknowns. This ordering of unknowns leads to a 2×2 "red-black partitioning" of the coefficient matrix; that is, a matrix of the form

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$$

with diagonal submatrices $A_{1,1}$ and $A_{2,2}$. The original linear system may require rearrangement in order to arrive at this form.

A mapping vector **IA(*)** is used to denote the starting locations of each of the contiguous blocks. The beginning of the linear block for row **I** is given by **IA(I)**, the end by **IA(I + 1) − 1**, and its length by **IA(I + 1) − IA(I)**. Thus, **IA(*)** will contain **N + 1** elements to accommodate a linear system of order **N**. The entries for each row may be stored in any order in the contiguous block for that row.

For example, the coefficient matrix

$$\begin{bmatrix} 11. & 0. & 0. & 14. & 15. \\ 0. & 22. & 0. & 0. & 0. \\ 0. & 0. & 33. & 0. & 0. \\ 14. & 0. & 0. & 44. & 45. \\ 15. & 0. & 0. & 45. & 55. \end{bmatrix}$$

would be represented in nonsymmetric sparse storage as

A(*) = [11., 14., 15., 22., 33., 14., 44., 45., 15., 45., 55.]

JA(*) = [1, 4, 5, 2, 3, 1, 4, 5, 1, 4, 5]

IA(*) = [1, 4, 5, 6, 9, 12]

and in symmetric sparse storage as

A(*) = [11., 14., 15., 22., 33., 44., 45., 55.]

JA(*) = [1, 4, 5, 2, 3, 4, 5, 5]

IA(*) = [1, 4, 5, 6, 8, 9]

3. USAGE

The user is expected to provide the coefficient matrix and the right-hand side of the linear system to be solved. The data structure for the matrix of the system is either the symmetric or nonsymmetric sparse storage format described in the previous section. An initial guess for the solution should be provided, if one is known; otherwise, it can be set to all zero values. A series of approximations for the solution are generated iteratively until the convergence criterion is satisfied. The algorithms are performed in two work-space arrays and some control over the algorithmic procedure can be obtained from switches in two parameter arrays.

There are seven main subroutines in ITPACK, each corresponding to an iterative method. They are

SUBROUTINE	METHOD
JCG( )	Jacobi Conjugate Gradient
JSI( )	Jacobi Semi-iteration
SOR( )	Successive Overrelaxation
SSORCG( )	Symmetric SOR Conjugate Gradient
SSORSI( )	Symmetric SOR Semi-iteration
RSCG( )	Reduced System Conjugate Gradient
RSSI( )	Reduced System Semi-iteration

and the calling sequence is

CALL ⟨method⟩ (N, IA, JA, A, RHS, U, IWKSP, NW, WKSP, IPARM, RPARM, IER)

where the parameters are defined in the following. Here "input" means that the subroutine expects the user to provide the necessary input data, and "output" means that the routine passes back information in the variable or array indicated. All parameters are linear arrays, except variables **N**, **NW**, and **IER**. Moreover, all parameters may be altered by the subroutine call, except variables **N** and **NW**. (See Section 7 for additional details.)

PARAMETER	DESCRIPTION
N	Order of the linear system. [integer; input]
IA(*)	Vector of length **N + 1** used in the sparse-matrix storage format. Contains the row pointers into **JA(*)** and **A(*)**. [integer array; input]
JA(*)	Vector of length **NZ** (defined in **A(*)** below) used in the sparse-matrix storage format. Contains the column numbers for the corresponding entries in **A(*)**. [integer array; input]
A(*)	Vector of length **NZ** used in the sparse-matrix storage format. Contains the nonzero entries of the coefficient matrix with positive diagonal elements. (**NZ** is the number of nonzero entries in the upper triangular part of the coefficient matrix when symmetric storage is used and is the total number of nonzeros when nonsymmetric storage is used.) [real array; input]
RHS(*)	Vector of length **N** containing the right-hand side of the linear system. [real array; input]
U(*)	Vector of length **N** containing the initial guess to the solution of the linear system on input and the latest approximate solution on output. [real array; input/output]
IWKSP(*)	Vector of length **3 * N** used for integer work space. When reindexing for red-black ordering, the first **N** locations contain on output the permutation vector for the red-black indexing, the next **N** locations contain its inverse, and the last **N** are used for integer work space.[2] [integer array; output]
NW	On input, **NW** is the available length for **WKSP(*)**. On output, **IPARM(8)** is the actual amount used (or needed). [integer; input]
WKSP(*)	Vector used for real working space whose length depends on the iterative method being used. Must be at least **NW** entries long. (See table near end of this section for required amount of work space for each method.) [real array]
IPARM(*)	Vector of length 12 used to initialize various integer and logical parameters. Default values may be set by calling subroutine **DFAULT( )** described below. On output, **IPARM(*)** contains the values of the parameters that were changed. (Further details to follow.) [integer array; input/output]
RPARM(*)	Vector of length 12 used to initialize various real parameters on input. Default values may be set by calling subroutine **DFAULT( )** described below. On output, **RPARM(*)** contains the final values of the parameters that were changed. (Further details given later in this section.) [real array; input/output]
IER	Error flag which is set to zero for normal convergence and to a nonzero integer when an error condition is present. (See table at end of section for meaning of nonzero values.) [integer; output]

The user may supply nondefault values for selected quantities in **IPARM(*)** and **RPARM(*)** by first executing

CALL DFAULT (IPARM, RPARM)

and then assigning the appropriate nondefault values before calling a solution module of ITPACK.

[2] For the red-black ordering, the **I**th entry of a permutation array **P( )** indicates the position **J** into which the **I**th unknown of the original system is being mapped; that is, if **P(I) = J**, then unknown **I** is mapped into position **J**. The **J**th entry of an inverse permutation array **IP( )** indicates the position **I** into which the **J**th unknown of the permuted system must be mapped to regain the original ordering, that is, **IP(J) = I**.

The iterative algorithms used in ITPACK are quite complicated and some knowledge of iterative methods is necessary to completely understand them. The interested reader should consult [4] and [6] for details. Important variables in this package which may change adaptively are **CME** (estimate of largest eigenvalue of the Jacobi matrix), **SME** (estimate of the smallest eigenvalue of the Jacobi matrix), **OMEGA** (overrelaxation parameter for the SOR and SSOR methods), **SPECR** (estimate of the spectral radius of the SSOR matrix), **BETAB** (estimate for the spectral radius of the matrix LU, where L and U are strictly lower and upper triangular matrices, respectively, such that the Jacobi matrix $B = L + U$).

The integer array **IPARM(*)** and real array **RPARM(*)** allow the user to control certain parameters which affect the performance of the iterative algorithms. Furthermore, these arrays allow the updated parameters from the automatic adaptive procedures to be communicated back to the user. The entries in **IPARM(*)** and **RPARM(*)** are

POSITION	DEFAULT	MEANING OR USAGE
IPARM(1)	100	**ITMAX:** Maximum number of iterations allowed. Reset on output to the number of iterations performed.
IPARM(2)	0	**LEVEL:** Control level of output. Each higher value provides additional information. [<0: no output on unit **IPARM(4)**; 0: fatal error messages only; 1: warning messages and minimum output; 2: reasonable summary (progress of algorithm); 3: parameter values and informative comments; 4: approximate solution after each iteration (primarily useful for debugging); 5: original system]
IPARM(3)	0	**IRESET**: Communication switch. [0: implies certain values of **IPARM(*)** and **RPARM(*)** will be overwritten to communicate parameters back to the user; otherwise: only **IPARM(1)** and **IPARM(8)** will be reset.]
IPARM(4)	6	**NOUT**: output unit number.
IPARM(5)	0	**ISYM**: Sparse-storage format switch. [0: symmetric sparse storage; 1: nonsymmetric sparse storage]
IPARM(6)	1	**IADAPT**: Adaptive switch. Determines whether certain parameters have been set by user or should be computed automatically in either a fully or partially adaptive sense. [0: fixed iterative parameters used for **SME, CME, OMEGA, SPECR,** and **BETAB** (nonadaptive); 1: fully adaptive procedures used for all parameters; 2: (SSOR methods only) **SPECR** determined adaptively and **CME, BETAB,** and **OMEGA** fixed; 3: (SSOR methods only) **BETAB** fixed and all other parameters determined adaptively] (See [4, 6] for details and **RPARM(I), I = 2, 3, 5, 6, 7**, for **CME, SME, OMEGA, SPECR, BETAB**, respectively. These parameters are

set by subroutine **DFAULT( )** or by the user.)

IPARM(7) 1 **ICASE**: Adaptive procedure case switch for JSI and SSOR methods. There are two strategies, called Case I and Case II, for doing the adaptive procedure. The choice of which case to select corresponds to knowledge of the eigenvalues of the Jacobi matrix B and their estimates.

[($\neq 2$) Case I: fixed **SME** $\leq m(B)$ (general case);
($= 2$) Case II: use when it is known that $|m(B)| \leq M(B)$]

The case switch determines how the estimates for **SME** and **CME** are recomputed adaptively. In Case I, **SME** is fixed throughout and should be less than or equal to $m(B)$. In Case II, **SME** is set to **−CME**, which may adaptively change. As far as the adaptive procedure is concerned, Case I is the most general case and should be specified in the absence of specific knowledge of the relationship between the eigenvalues and their estimates. An example when Case II is appropriate occurs when the Jacobi matrix has Property A, since $m(B) = -M(B)$.[3] Also, if A is an L-matrix, then for the Jacobi matrix, we have $|m(B)| \leq M(B)$ and **SME** is always **−CME** (Case II).[4] Selecting the correct case may increase the rate of convergence of the iterative method. (See [6] for additional discussion on Cases I and II. Also, see **RPARM(I), I = 2, 3** for **CME**, **SME**, respectively.)

IPARM(8) 0 **NWKSP**: Amount of work space used. Used for output only. If **ITMAX** is set to a value just over the actual number of iterations necessary for convergence, the amount of memory for **WKSP(*)** can be reduced to just over the value returned here. This may be done when rerunning a problem, for example.

IPARM(9) −1 **NB**: Red-black ordering switch. On output, if reindexing is done, **NB** is set to the order of the black subsystem.

[For RS methods,
<0: compute red--black indexing and permute system;
≥0: skip indexing—system already in red–black form;
For other methods,
<0: skip indexing—system already in desired form;
≥0: compute red--black indexing and permute system]

[3] A matrix has Property A if and only if it is a diagonal matrix, or else there exists a rearrangement of the rows and corresponding columns of the matrix which corresponds to a red–black partitioning.

[4] An L-matrix has positive diagonal elements and nonpositive off-diagonal elements.

A negative integer value for **IPARM(9)** causes the equations to be handled in the most general way appropriate for the solution method being used. For methods other than RS methods this is the "natural order," while for RS methods it is the "red-black order." A nonnegative value produces a red-black permutation for all methods except for the RS methods, which are assumed to be in red-black order with the order of the black subsystem **NB** given. If reindexing is performed, **IPARM(9)** will contain the order of the black subsystem on output.

IPARM(10) 0 **IREMOVE**: Switch for effectively removing rows and columns when the diagonal entry is extremely large compared to the nonzero off-diagonal entries in that row. (See **RPARM(8)** for additional details.)

[0: not done; otherwise: test done]

IPARM(11) 0 **ITIME**: Timing switch.

[0: time method; otherwise: not done]

IPARM(12) 0 **IDGTS**: Error analysis switch. An analysis of final computed solution to determine accuracy.

[<0: skip error analysis
0: compute **DIGIT1** and **DIGIT2** and store in **RPARM(I), I = 11, 12, respectively;**
1: print **DIGIT1** and **DIGIT2**;
2: print final approximate solution vector;
3: print final approximate residual vector;
4: print both solution and residual vectors;
otherwise: no printing]

(If **LEVEL** $\leq$ 0, no printing is done. See **RPARM(I), I = 11, 12,** for details on **DIGIT1** and **DIGIT2**.)

ENTRY DEFAULT	DESCRIPTION
RPARM(1) 5.E−6	**ZETA**: Stopping criterion or approximate relative accuracy desired in the final computer solution. If the method does not converge in **IPARM(1)** iterations, **RPARM(1)** is reset to an estimate of the relative accuracy achieved. The stopping criterion is a test of whether **ZETA** is greater than the ratio of the Euclidean norm of the pseudo-residual vector and the norm of the current iteration vector times a constant involving an eigenvalue estimate. (See [4, 6] for details.)
RPARM(2) 0.	**CME**: Estimate of largest eigenvalue of Jacobi matrix. Changes to new estimate if adaptive procedure is used. **CME** $\leq M(B)$.
RPARM(3) 0.	**SME**: Estimate of smallest eigenvalue of Jacobi matrix for JSI method. In Case I, **SME** is fixed throughout at a value $\leq m(B)$. In Case II, **SME** is always set to **−CME** with **CME** changing in the adaptive procedure. (See **IPARM(7)** for definitions of Cases I and II.)

RPARM(4) .75 — **FF**: Adaptive procedure damping factor. Values in the interval (0., 1.] with 1. causing the most frequent parameter changes when the fully adaptive switch **IPARM(6) = 1** is used.

RPARM(5) 1. — **OMEGA**: Overrelaxation parameter for SOR and SSOR methods. If method is fully adaptive, **OMEGA** changes.

RPARM(6) 0. — **SPECR**: Estimated spectral radius for SSOR matrix. If method is adaptive, **SPECR** changes.

RPARM(7) .25 — **BETAB**: Estimate for the spectral radius of matrix LU used in SSOR methods. **BETAB** may change depending on the adaptive switch **IPARM(6)**. The matrix L is the strictly lower triangular part of the Jacobi matrix and U the strictly upper triangular part. When the spectral radius of LU is less than or equal to $\frac{1}{4}$, the "SSOR condition" is satisfied for some problems, provided one uses the natural ordering. (See [4, 5, 19] for additional details.)

RPARM(8) 100. * SRELPR — **TOL**: Tolerance factor near machine relative precision, **SRELPR**. In each row, if all nonzero off-diagonal row entries are less than **TOL** times the value of the diagonal entry, then this row and corresponding column are essentially removed from the system. This is done by setting the nonzero off-diagonal elements in the row and corresponding column to zero, replacing the diagonal element with 1, and adjusting the elements on the right-hand side of the system so that the new system is equivalent to the original one.[5] If the diagonal entry is the only nonzero element in a row and is not greater than the reciprocal of **TOL**, then no elimination is done. This procedure is useful for linear systems arising from finite-element discretizations of PDEs in which Dirichlet boundary conditions are handled by giving the diagonal values in the linear system extremely large values. (The installer of this package should set the value of **SRELPR**. See comments in subroutine **DFAULT( )** for additional details.)

RPARM(9) 0. — **TIME1**: Total time in seconds from beginning of iterative algorithm until convergence. (A machine-dependent subprogram call for returning the time in seconds is provided by the installer of this package.)

RPARM(10) 0. — **TIME2**: Total time in seconds for entire call.

RPARM(11) 0. — **DIGIT1**: Approximate number of digits using the estimated relative error with the final approximate solution. Computed as the negative of the logarithm base 10 of the final value of the stopping test. (See details below or [6].)

RPARM(12) 0. — **DIGIT2**: Approximate number of digits using the estimated relative residual with the final approximate solution. Computed as the negative of the logarithm base 10 of the ratio of the two norm of the residual vector and the two norm of the right-hand-side vector. This estimate is related to the condition number of the original linear system and therefore it will not be accurate if the system is ill-conditioned. (See details below or [6].)

[5] If the row and column corresponding to diagonal entry coef(i, i) are to be eliminated, then the right-hand side is adjusted to rhs(i) = rhs(i)/coef(i, i) and rhs(j) = rhs(j) − rhs(i) × coef(i, j) for $j \neq i$.

DIGIT1 is determined from the actual stopping test computed on the final iteration, whereas **DIGIT2** is based on the computed residual vector using the final approximate solution after the algorithm has converged. If these values differ greatly, then either the stopping test has not worked successfully or the original system is ill-conditioned. (See [6] for additional details.)

For storage of certain intermediate results, the solution modules require a real vector **WKSP(*)** and a corresponding variable **NW** indicating the available space. The length of the work-space array varies with each solution module and the maximum amount needed is given in the following table.

SOLUTION MODULE	MAXIMUM LENGTH OF WKSP(*)
JCG( )	**4 * N + NCG**
JSI( )	**2 * N**
SOR( )	**N**
SSORCG( )	**6 * N + NCG**
SSORSI( )	**5 * N**
RSCG( )	**N + 3 * NB + NCG**
RSSI( )	**N + NB**

where value of **NCG** is **2 * IPARM(1)** for symmetric sparse storage and **4 * IPARM(1)** for nonsymmetric sparse storage. It should be noted that the actual amount of work space used may be somewhat less than these upper limits since some of the latter are dependent on the maximum number of iterations allowed, **ITMAX**, stored in **IPARM(1)**. Clearly, the array **WKSP(*)** must be dimensioned to at least the value of **NW**.

Nonzero integer values of the error flag **IER** indicate that an error condition was detected. These values are listed below according to their numerical value and to the name of the routine in which the flag was set.

ERROR FLAG	MEANING
IER = 0,	Normal convergence obtained.
= 1 + Mth,	Invalid order of system, **N**
= 2 + Mth,	Work-space array **WKSP(*)** not large enough—**IPARM(8)** set to amount of required work space, **NW**.
= 3 + Mth,	Failure to converge in **IPARM(1)** iterations—**RPARM(1)** reset to last stopping value computed.
= 4 + Mth,	Invalid order of black subsystem, **NB**.
= 101,	Diagonal element not positive.
= 102,	No diagonal entry in a row.
= 201,	Red-black indexing not possible.
= 301,	No entry in row of original matrix.
= 302,	No entry in row of permuted matrix.
= 303,	Sorting error in row of permuted matrix.
= 401,	Diagonal element not positive.
= 402,	No diagonal entry in a row.
= 501,	Failure to converge in **ITMAX** evaluations.
= 502,	Function does not change sign at endpoints.
= 601,	Successive iterates not monotone increasing.

JCG( ), JSI( ), SOR( ), SSORCG( ), SSORSI( ), RSCG( ), RSSI( ) assign values to Mth of 10, 20, 30, 40, 50, 60, 70, respectively. **SBELM( ), PRBNDX( ), PERMAT( ), SCAL( ), ZBRENT( ), EQRT1S( )** are subroutines with error flags in the 100s, 200s, 300s, 400s, 500s, 600s, respectively. These routines perform the following functions: **SBELM( )** removes rows and columns, **PRBNDX( )** determines the red-black indexing, **SCAL( )** scales the system, **ZBRENT( )** is a modified IMSL routine for computing a zero of a function which changes sign in

a given interval, **EQRT1S( )** is a modified IMSL routine for computing the largest eigenvalue of a symmetric tridiagonal matrix.[6]

4. USER-ORIENTED MODULES

The array **U(*)** should contain an initial approximation to the solution of the linear system before any ITPACK module is called. If the user has no information for making such a guess, then the zero vector may be used as the starting vector. The subroutine **VFILL( )** can be used to fill a vector with a constant:

CALL VFILL (N, U, VAL)

fills array **U(*)** of length **N** with value **VAL** in each entry.

To aid the user in using the iterative methods of ITPACK, four modules for constructing the sparse-matrix storage arrays are included. The modules are

SBINI( ) called at the beginning to initialize the arrays **IA(*)**, **JA(*)**, **A(*)**, and **IWORK(*)**;

SBSIJ( ) called repeatedly to set the individual entries in the matrix and build a link-list representation of the matrix structure;

SBEND( ) called at the end to restructure the link list into final sparse storage form;

SBAGN( ) called to return again to the link-list representation if **SBEND( )** has been called but additional elements are to be added or modified.

These modules are described below.

(a) Initialization:

CALL SBINI (N, NZ, IA, JA, A, IWORK)

Initializes **IA(*)**, **JA(*)**, **A(*)**, and **IWORK(*)** for a system of order **N**. **IA(*)**, **JA(*)**, and **IWORK(*)** are integer arrays of length at least **N + 1**, **NZ**, and **NZ**, respectively. **A(*)** is a real array of length at least **NZ**.

(b) Set individual entries:

CALL SBSIJ (N, NZ, IA, JA, A, IWORK, I, J, VAL, MODE, LEVEL, NOUT, IER)

Inserts the value, **VAL**, of the **(I, J)** entry of the user's matrix into the link-list representation for that matrix. When using symmetric sparse storage, **J** must be greater than or equal to **I**. If the **(I, J)** entry has already been set, then **MODE** specifies the way in which the entry is to be treated:

MODE < 0, Current entry value is left as is;
$= 0$, Current entry value is reset to **VAL**;
> 0, **VAL** is added to the current entry value.

If **LEVEL** is less than 0, **SBSIJ( )** causes no printing. If **LEVEL** is 0, fatal error messages are written to output unit number **NOUT**; and if **LEVEL** is 1 or greater, a message is printed when **SBSIJ( )** encounters a value it has already set with the value being reset according to the value of **MODE**. **IER** is an error parameter and returns values of

ERROR FLAG	MEANING
IER = 0,	New **(I, J)** entry established.
= 700,	**(I, J)** entry already set—reset according to **MODE**.
= 701,	Improper values for either **I** or **J**.
= 702,	**NZ** too small—no room for new entry.

[6] International Mathematical and Statistical Libraries, Inc., Sixth Floor NBC Bldg., 7500 Bellaire Blvd., Houston, TX 77036.

(c) Finalization:

CALL SBEND (N, NZ, IA, JA, A, IWORK)

Restructures the link-list data structure built by **SBINI( )** and **SBSIJ( )** into the final data structure required by ITPACK.

(d) Undo finalization:

CALL SBAGN (N, NZ, IA, JA, A, IWORK, LEVEL, NOUT, IER)

Returns to link-list representation for modification or addition of elements to the system. Repeated calls to **SBSIJ( )** can then be made followed by a single call to **SBEND( )** to close out the sparse-matrix representation. If **LEVEL** is less than 0, no printing is done, and if **LEVEL** is 0 or greater, fatal error information is written to output unit number **NOUT**. **IER** is an error flag indicating

ERROR FLAG	MEANING
IER = 0,	Successful completion.
= 703,	**NZ** too small—no room for new entry.

Note that **SBINI( )** should not be called after **SBAGN( )** is called, since it would destroy the previous data.

5. EXAMPLES

Given a linear system $Au = b$ with

$$A = \begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix}, \qquad b = \begin{bmatrix} 6 \\ 0 \\ 0 \\ 6 \end{bmatrix}$$

a program to solve this problem with an initial guess of $u^T = (0, 0, 0, 0)$ using **JCG( )** with symmetric sparse storage and printing the final approximate solution vector follows.

```
      INTEGER IA(5), JA(8), IPARM(12), IWKSP(12)
      REAL A(8), RHS(4), U(4), WKSP(24), RPARM(12)
      DATA A(1), A(2), A(3), A(4)/4.0, -1.0, -1.0, 4.0/
      DATA A(5), A(6), A(7), A(8)/-1.0, 4.0, -1.0, 4.0/
      DATA JA(1), JA(2), JA(3), JA(4)/1, 2, 3, 2/
      DATA JA(5), JA(6), JA(7), JA(8)/4, 3, 4, 4/
      DATA IA(1), IA(2), IA(3), IA(4), IA(5)/1, 4, 6, 8, 9/
      DATA RHS(1), RHS(2), RHS(3), RHS(4)/6.0, 0.0, 0.0, 6.0/
      DATA N/4/, NW/24/, ITMAX/4/, LEVEL/1/, IDGTS/2/
C
      CALL DFAULT (IPARM, RPARM)
      IPARM(1) = ITMAX
      IPARM(2) = LEVEL
      IPARM(12) = IDGTS
      CALL VFILL (N, U, 0.E0)
      CALL JCG (N, IA, JA, A, RHS, U, IWKSP, NW, WKSP, IPARM, RPARM, IER)
      STOP
      END
```

The output for this run would be

```
BEGINNING OF ITPACK SOLUTION MODULE JCG
JCG HAS CONVERGED IN      2 ITERATIONS.
      APPROX. NO. OF DIGITS (EST. REL. ERROR) = 14.6 (DIGIT1)
 APPROX. NO. OF DIGITS (EST. REL. RESIDUAL) = 14.3 (DIGIT2)
 SOLUTION  VECTOR.
          1              2              3              4
 ----------------------------------------------------------
  2.00000E+00    1.00000E+00    1.00000E+00    2.00000E+00
```

Textbook methods such as the Jacobi (J), Gauss-Seidel (GS), successive overrelaxation (SOR—fixed relaxation factor omega), symmetric successive

Table I

Method	Use	Parameters
J	**JSI( )**	**IPARM(6) = 0, IPARM(7) = 2**
GS	**SOR( )**	**IPARM(6) = 0**
SOR—fixed omega	**SOR( )**	**IPARM(6) = 0, RPARM(5) = OMEGA**
SSOR—fixed omega	**SSORSI( )**	**IPARM(6) = 0, RPARM(5) = OMEGA**
RS	**RSSI( )**	**IPARM(6) = 0**

overrelaxation (SSOR—fixed relaxation factor omega), and the RS method can be obtained from this package by resetting appropriate parameters after the subroutine **DFAULT( )** is called but before ITPACK routines are called (see Table I). These methods were not included as separate routines because they are usually slower than the accelerated methods included in this package.

On the black unknowns, the cyclic Chebyshev semi-iterative (CCSI) method of Golub and Varga [2] gives the same result as the RSSI method. The CCSI and RSSI methods converge at the same rate, and each of them converges twice as fast as the JSI method. This is a theoretical result [6] and does not count the time involved in establishing the red-black indexing and the red-black partitioned system. Similarly, the cyclic conjugate gradient (CCG) method with respect to the black unknowns, considered by Reid [16] (see also Hageman and Young [6]), gives the same results as the RSCG method. Also, the CCG and the RSCG methods converge at the same rate, and each of them converges, theoretically, exactly twice as fast as the JCG method. Hence, the accelerated RS methods are preferable to the accelerated J methods when using a red-black indexing.

6. NUMERICAL RESULTS

The iterative algorithms in ITPACK have been tested over a wide class of matrix problems arising from elliptic partial differential equations with Dirichlet, Neumann, and mixed boundary conditions on arbitrary two-dimensional regions (including cracks and holes) and on rectangular three-dimensional regions [1]. Both finite-difference and finite-element procedures have been employed to obtain the linear systems. The two sample problems presented here, while simple to pose, are representative of the behavior of the ITPACK routines for more complex problems. The iterative algorithms make no use of the constant coefficients in these two problems or of the particular structure of the resulting linear system. Because the ITPACK code is not tailored to any particular class of partial differential equations or discretization procedure, but rather to sparse linear systems, it is felt that the package can be used to solve a wider class of problems.

We now consider two simple partial differential equations that, when discretized by finite-difference methods, give rise to large sparse linear systems. We obtain the solution of each of these systems by the seven algorithms in ITPACK 2C. These numerical results should aid the user of ITPACK in determining the amount of time required when solving more complicated sparse systems. However, one should not interpret these execution times as conclusive by themselves. Variances introduced by different compilers, computer systems, and timing functions can sometimes be significant. Moreover, the number of iterations required by an iterative method is dependent on the problem being solved, the initial estimate for the solution, the parameter estimates used, and the relative accuracy requested in the stopping criterion **RPARM(1)**. These tests were run on the CDC CYBER 170/750 at the University of Texas with the FTN 4.8 compiler (OPT = 2).

To obtain representative sparse linear systems, we discretize the following two self-adjoint elliptic partial differential equations in a region with prescribed conditions on the boundary. Here u_{xx}, u_{yy}, u_{zz} are partial derivatives and du/dn is the derivative in the normal direction.

Table II. Number of Iterations and Execution Times for Problem (1) Using Adaptive and Nonadaptive Procedures (Nonadaptive Data in Parentheses)

Routine	Ordering	Iterations	Iteration time	Total time
JCG( )	Natural	61 (61)	0.250 (0.247)	0.281 (0.271)
	Red-black	61 (61)	0.232 (0.246)	0.402 (0.413)
JSI( )	Natural	108 (95)	0.408 (0.344)	0.439 (0.375)
	Red-black	108 (95)	0.393 (0.332)	0.569 (0.498)
SOR( )	Natural	72 (54)	0.356 (0.280)	0.368 (0.307)
	Red-black	65 (47)	0.311 (0.224)	0.469 (0.411)
SSORCG( )	Natural	17 (13)	0.232 (0.173)	0.264 (0.185)
SSORSI( )	Natural	23 (22)	0.242 (0.213)	0.273 (0.244)
RSCG( )	Red-black	31 (31)	0.104 (0.117)	0.269 (0.297)
RSSI( )	Red-black	60 (48)	0.207 (0.166)	0.358 (0.344)

Table III. Number of Iterations and Execution Times for Problem (2) Using Adaptive and Nonadaptive Procedures (Nonadaptive Data in Parentheses)

Routine	Ordering	Iterations	Iteration time	Total time
JCG( )	Natural	28 (28)	0.092 (0.090)	0.107 (0.090)
	Red-black	28 (28)	0.079 (0.074)	0.191 (0.202)
JSI( )	Natural	64 (54)	0.166 (0.136)	0.196 (0.152)
	Red-black	64 (54)	0.160 (0.130)	0.268 (0.266)
SOR( )	Natural	42 (29)	0.139 (0.095)	0.150 (0.110)
	Red-black	38 (29)	0.124 (0.097)	0.236 (0.231)
SSORCG( )	Natural	15 (11)	0.136 (0.097)	0.167 (0.111)
SSORSI( )	Natural	19 (15)	0.138 (0.101)	0.153 (0.117)
RSCG( )	Red-black	15 (15)	0.032 (0.051)	0.150 (0.169)
RSSI( )	Red-black	31 (27)	0.075 (0.064)	0.186 (0.196)

$$\begin{cases} u_{xx} + 2u_{yy} = 0, & (x, y) \quad \text{in} \quad S = (0, 1) \times (0, 1) \\ u = 1 + xy, & (x, y) \quad \text{on boundary of} \quad S. \end{cases} \tag{1}$$

Using the standard 5-point symmetric finite-difference operator with $h = \frac{1}{20}$, we obtain a sparse linear system with 1729 nonzero elements and 361 unknowns.

$$\begin{cases} u_{xx} + 2u_{yy} + 3u_{zz} = 0, & (x, y, z) \quad \text{in} \quad C = (0, 1) \times (0, 1) \times (0, 1) \\ \text{On boundary of } C: & \\ \quad \begin{cases} u = 1, & (0, y, z), (x, 0, z), \quad \text{or} \quad (x, y, 0) \\ \dfrac{du}{dn} = \begin{cases} yz(1 + yz), & (1, y, z) \\ xz(1 + xz), & (x, 1, z) \\ xy(1 + xy), & (x, y, 1). \end{cases} \end{cases} \end{cases} \tag{2}$$

Using the standard 7-point symmetric finite difference operator with $h = \frac{1}{7}$, we obtain a sparse linear system with 1296 nonzero elements and 216 unknowns.

Tables II and III display the number of iterations and execution times (in seconds) for the seven methods in ITPACK 2C for the linear systems corresponding to problems (1) and (2), respectively, using symmetric sparse storage. Both the time for the iteration algorithm and the total time for the subroutine call are given. The stopping criterion was set to **5.0E−6**. To illustrate how effective the adaptive procedures are, we have included in these tables the number of iterations and the time when the optimum iteration parameters were used with no adaptive procedures.

Values corresponding to the red-black ordering with the SSOR methods are omitted from the tables since it is known that these methods are ineffective with

this ordering. Since the RS methods are defined for only the red–black ordering, the table entries for these methods with the natural ordering are not included.

7. NOTES ON USE

Before an iterative algorithm is called to solve a linear system, the values in the array **A(*)** are permuted and scaled. Afterward, these values are unpermuted and unscaled. Consequently, the values in arrays **A(*)** and **RHS(*)** may change slightly due to round-off errors in the computer arithmetic. Moreover, since entries in each row of the linear system may be stored in any order within a contiguous block of data, the locations of elements of **A(*)** and of corresponding ones in **JA(*)** may change from those given before the permuting and unpermuting was done. The same linear system is defined by the arrays **A(*)**, **JA(*)**, and **IA(*)**, whether or not corresponding elements in **A(*)** and **JA(*)** have changed locations within contiguous blocks.

Scaling of the linear system is done as follows to reduce the number of arithmetic operations. The diagonal entries of the linear system are checked for positivity and are moved to the first **N** locations of the array **A(*)**. The nonzero off-diagonal entries of the linear system $Au = b$ are scaled. The scaling involves the diagonal matrix $D^{1/2}$ of square roots of the diagonal entries of the linear system, that is,

$$(D^{-1/2}AD^{-1/2})(D^{1/2}u) = (D^{-1/2}b).$$

The algorithms iterate until convergence is reached based on the relative accuracy requested via the stopping criterion set in **RPARM(1)** for the scaled solution vector $(D^{1/2}u)$. Unscaling solves for u and returns the linear system to its original form subject to round-off errors in the arithmetic and to possible movement of entries within contiguous blocks of data.

When requested, a red–black permutation of the data will be done before and after the iterative algorithm is called. Otherwise, the linear system is used in the order it is given, which we call the "natural ordering."

The successive overrelaxation (SOR) method has been shown to be more effective with the red–black ordering than with the natural ordering for some problems [19]. In the SOR algorithm, the first iteration uses **OMEGA = 1.0** and the stopping criterion is set to a large value so that at least one Gauss–Seidel iteration is performed before an approximate value for the optimum relaxation parameter is computed.

Optional features of this package are red–black ordering, effective removal of rows and columns when the diagonal entry is extremely large, and error analysis. In the event that one is not using some of these options and needs additional memory space for a very large linear system, the relevant subroutines that can be replaced with dummy subroutines are as follows: red–black ordering [**PRBNDX( )**, **PERMAT( )**, **PERVEC( )**, **QSORT( )**], removal of rows [**SBELM( )**] , error analysis [**PERROR( )**].

The timing routines **ITICK( )** and **ITOCK( )** should call a local system routine so that they return the run time in milliseconds. **ITICK( )** is called at the beginning of a timing interval and **ITOCK( )** at the end.

The value of the machine relative precision is contained in the variable **SRELPR**, which is set in the subroutine **DFAULT( )** and in the test program. This and other default values may be permanently changed when the code is installed by changing their values in the subroutine **DFAULT( )**. **SRELPR** must be changed when moving the code to another computer. If the installer of this package does not know its value, an approximate value can be determined from a simple FORTRAN program given in the comment statements of subroutine **DFAULT( )**.

Since the amount of precision may change from computer to computer, the relative accuracy requested in the stopping criterion **ZETA** must not be less than about **500.** times the machine relative precision **SRELPR**. If a value of **ZETA** is

requested that is too small, then the code resets it to this value. The current default value for **ZETA**, **5.0E−6**, is set by the routine **DFAULT( )** into **RPARM(1)**.

The distribution tape contains the ITPACK 2C software package of 71 subprograms and a testing program **MAIN( )** together with its 27 subprograms. The routines **DFAULT( )**, **ITICK( )**, **ITOCK( )** in ITPACK and the program **MAIN( )** are the only ones requiring editing by the installer of the package. ITPACK can be made into a compiled program library, although not all of it would normally be used in a particular application.

8. ITPACK HISTORY

The 2C version of the ITPACK codes described here is the result of several years of research and development. The development of ITPACK began in the early 1970s when Professor Garrett Birkhoff suggested that general-purpose software for solving linear systems should be developed for iterative methods as well as for direct methods. Initially, prototype programs were written based on preliminary iterative algorithms involving adaptive selection of parameters and automatic stopping procedures. These programs were tested on a large set of elliptic partial differential equations over domains compatible with the subroutine **REGION( )** [8], which superimposed a square grid over the domain. These routines were designed for solving self-adjoint elliptic partial differential equations. Next a preliminary version of ITPACK was coded in standard FORTRAN. The ITPACK routines used iterative algorithms that were refined from the prototype programs. However, these routines were designed to solve large sparse linear systems of algebraic equations instead of partial differential equations. The use of three interchangeable symmetric sparse storage modes in ITPACK 1.0 [3] allowed for great flexibility and made it possible to solve a wider class of problems than the prototype programs and to study different storage modes for iterative methods. The next version, ITPACK 2.0 [4], was significantly faster than its predecessor, since it was restricted to allow only one sparse symmetric storage format. Most of the iterative algorithms utilized in the 2.0 version of this package assume that the coefficient matrix of the linear system is symmetric positive definite. As with many packages, the need to handle a slightly larger class of problems, namely, nearly symmetric systems, soon became evident. This required adapting the routines to allow a switch for either a symmetric or nonsymmetric storage mode in ITPACK 2A [5]. Moreover, a modification of the conjugate gradient algorithms was developed to handle nearly symmetric systems [13]. ITPACK has been improved in the 2B version [11] by (a) writing more efficient versions of several key subroutines, (b) incorporating basic linear algebra subprograms (BLAS) [15], and (c) improving the user interface with better printing and documentation. Some additional improvements and corrections were made in the 2C version. The algorithms in ITPACK are not guaranteed to converge for all linear systems, but have been shown to work successfully for a large number of symmetric and nonsymmetric systems that arise from solving elliptic partial differential equations [1, 14].

The numerical algorithms in ITPACK 2C correspond to those described in the appendix of technical report [4] and outlined in the book [6]. In particular, the SOR code is based on an algorithm suggested to us by L. Hageman. Various other algorithms exist for iterative methods. For example, S. Eisenstat has an implementation of the symmetric successive overrelaxation preconditioned conjugate gradient procedure.[7]

Modules based on the seven iterative routines in ITPACK have been incorporated into the elliptic partial differential equation solving package ELLPACK [17] together with all the necessary translation routines needed. The user-oriented modules described in Section 4 are not in ELLPACK. Moreover, if the ELLPACK

[7] Private communication.

system is not being used to generate the linear system for ITPACK, it is recommended that ITPACK be used as a stand-alone package apart from ELLPACK.

ACKNOWLEDGMENTS

The authors wish to thank the referee for carefully going through the code and documentation for several different versions. Test runs were made on a variety of computer systems and helpful suggestions were made by R. Boisvert, W. Coughran, J. Dongarra, W. Dyksen, S. Eisenstat, S. Fillebrown, P. Gaffney, W. Gordon, R. Hanson, R. Lynch, J. Rice, B. Ward, and others. These suggestions and comments together with those of the referee have resulted in an improved software package. ITPACK has been tested on the following computing machines: CDC 6400, 6500, 6600, 7600; CYBER 170/750, 203, 205; CRAY 1; DEC 10, 20; PDP 10; VAX 11/750, 11/780; IBM 195, 370/158, 3033; PRIME 400, 750; and others.

REFERENCES

(Note. References [9, 10, 18] are not cited in the text.)

1. EISENSTAT, S., GEORGE, A., GRIMES, R., KINCAID, D., AND SHERMAN, A. Some comparisons of software packages for large sparse linear systems. In *Advances in Computer Methods for Partial Differential Equations III*, R. Vichnevetsky and R. Stepleman (Eds.), pub. IMACS, Dep. of Computer Science, Rutgers Univ., New Brunswick, N. J., 1979, pp. 98–106.
2. GOLUB, G., AND VARGA, R. Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second-order Richardson iterative methods, parts I and II. *Numer. Math. 3* (1961), 147–168.
3. GRIMES, R., KINCAID, D., MACGREGOR, W., AND YOUNG, D. ITPACK report: Adaptive iterative algorithms using symmetric sparse storage. CNA-139, Center for Numerical Analysis, Univ. of Texas, Austin, Aug. 1978.
4. GRIMES, R., KINCAID, D., AND YOUNG, D. ITPACK 2.0 user's guide. CNA-150, Center for Numerical Analysis, Univ. of Texas, Austin, Aug. 1979.
5. GRIMES, R., KINCAID, D., AND YOUNG, D. ITPACK 2A: A Fortran implementation of adaptive accelerated iterative methods for solving large sparse linear systems. CNA-164, Center for Numerical Analysis, Univ. of Texas, Austin, Oct. 1980.
6. HAGEMAN, L., AND YOUNG, D. *Applied Iterative Methods.* Academic Press, New York, 1981.
7. HAYES, L., AND YOUNG, D. The accelerated SSOR method for solving large linear systems: Preliminary report. CNA-123, Center for Numerical Analysis, Univ. of Texas, Austin, May 1977.
8. KINCAID, D., AND GRIMES, R. Numerical studies of several adaptive iterative algorithms. CNA-126, Center for Numerical Analysis, Univ. of Texas, Austin, Aug. 1977.
9. KINCAID, D., GRIMES, R., MACGREGOR, W., AND YOUNG, D. ITPACK—Adaptive iterative algorithms using symmetric sparse storage. In *Proc. Symp. Reservoir Simulation*, Soc. Petroleum Eng. of AIME, Dallas February 1979, pp. 151–160.
10. KINCAID, D., GRIMES, R., AND YOUNG, D. The use of iterative methods for solving large sparse PDE-related linear systems. In *Mathematics and Computers in Simulation XXI*, Elsevier North-Holland, New York, 1979, pp. 368–375.
11. KINCAID, D., GRIMES, R., RESPESS, J., AND YOUNG, D. ITPACK 2B: A Fortran implementation of adaptive accelerated iterative methods for solving large sparse linear systems. CNA-173, Center for Numerical Analysis, Univ. of Texas, Austin, Sept. 1981.
12. KINCAID, D., AND YOUNG, D. Survey of iterative methods. In *Encyclopedia of Computer Sciences and Technology*, vol. 13, J. Belzer, A. Holzman, and A. Kent (Eds.), Marcel Dekker, New York, 1979, pp. 354–391.
13. KINCAID, D., AND YOUNG, D. Adapting iterative algorithms developed for symmetric systems to nonsymmetric systems. In *Elliptic Problem Solvers*, M. Schultz (Ed.), Academic Press, New York, 1981, p. 353–359.
14. KINCAID, D. Acceleration parameters for a symmetric successive overrelaxation conjugate gradient method for nonsymmetric systems. In *Advances in Computer Methods for Partial Differential Equations IV*, R. Vichnevetsky and R. Stepleman (Eds.), IMACS, Dep. of Computer Science, Rutgers Univ., New Brunswick, N. J., 1981, pp. 294–299.
15. LAWSON, C. L., HANSON, R. J., KINCAID, D. R., AND KROGH, F. T. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Softw. 5*, 3 (Sept. 1979), 308–323.
16. REID, J. The use of conjugate gradients for systems of linear equations possessing property A. *SIAM J. Numer. Anal. 9*, (1972), 325–332.
17. RICE, J. ELLPACK user's guide. CSD-TR 372, Computer Science Dep., Purdue Univ., West Lafayette, Ind., July 1981.
18. YOUNG, D., AND KINCAID, D. The ITPACK package for large sparse linear systems. In *Elliptic Problem Solvers*, M. Schultz (Ed.), Academic Press, New York, 1981, pp. 163–185.
19. YOUNG, D. *Iterative Solution of Large Linear Systems.* Academic Press, New York, 1971.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service.]

```
      SUBROUTINE JCG(NN,IA,JA,A,RHS,U,IWKSP,NW,WKSP,                    JCG ØØ1Ø
     A               IPARM,RPARM,IERR)                                  JCG ØØ2Ø
C                                                                       JCG ØØ3Ø
C     ITPACK 2C MAIN SUBROUTINE  JCG  (JACOBI CONJUGATE GRADIENT)       JCG ØØ4Ø
C     EACH OF THE MAIN SUBROUTINES:                                     JCG ØØ5Ø
C           JCG, JSI, SOR, SSORCG, SSORSI, RSCG, RSSI                   JCG ØØ6Ø
C     CAN BE USED INDEPENDENTLY OF THE OTHERS                           JCG ØØ7Ø
C                                                                       JCG ØØ8Ø
C ... FUNCTION:                                                         JCG ØØ9Ø
C                                                                       JCG Ø1ØØ
C          THIS SUBROUTINE, JCG, DRIVES THE JACOBI CONJUGATE            JCG Ø11Ø
C          GRADIENT ALGORITHM.                                          JCG Ø12Ø
C                                                                       JCG Ø13Ø
C ... PARAMETER LIST:                                                   JCG Ø14Ø
C                                                                       JCG Ø15Ø
C          N       INPUT INTEGER.  DIMENSION OF THE MATRIX. (= NN)      JCG Ø16Ø
C          IA,JA   INPUT INTEGER VECTORS.  THE TWO INTEGER ARRAYS OF    JCG Ø17Ø
C                  THE SPARSE MATRIX REPRESENTATION.                    JCG Ø18Ø
C          A       INPUT REAL VECTOR.  THE REAL ARRAY OF THE SPARSE     JCG Ø19Ø
C                  MATRIX REPRESENTATION.                               JCG Ø2ØØ
C          RHS     INPUT REAL VECTOR.  CONTAINS THE RIGHT HAND SIDE     JCG Ø21Ø
C                  OF THE MATRIX PROBLEM.                               JCG Ø22Ø
C          U       INPUT/OUTPUT REAL VECTOR.  ON INPUT, U CONTAINS THE  JCG Ø23Ø
C                  INITIAL GUESS TO THE SOLUTION. ON OUTPUT, IT CONTAINS JCG Ø24Ø
C                  THE LATEST ESTIMATE TO THE SOLUTION.                 JCG Ø25Ø
C          IWKSP   INTEGER VECTOR WORKSPACE OF LENGTH 3*N               JCG Ø26Ø
C          NW      INPUT INTEGER.  LENGTH OF AVAILABLE WKSP.  ON OUTPUT, JCG Ø27Ø
C                  IPARM(8) IS AMOUNT USED.                             JCG Ø28Ø
C          WKSP    REAL VECTOR USED FOR WORKING SPACE.  JACOBI CONJUGATE JCG Ø29Ø
C                  GRADIENT NEEDS THIS TO BE IN LENGTH AT LEAST         JCG Ø3ØØ
C                  4*N + 2*ITMAX,  IF ISYM = Ø  (SYMMETRIC STORAGE)     JCG Ø31Ø
C                  4*N + 4*ITMAX,  IF ISYM = 1  (NONSYMMETRIC STORAGE)  JCG Ø32Ø
C                  HERE ITMAX = IPARM(1) AND ISYM = IPARM(5)            JCG Ø33Ø
C                  (ITMAX IS THE MAXIMUM ALLOWABLE NUMBER OF ITERATIONS) JCG Ø34Ø
C          IPARM   INTEGER VECTOR OF LENGTH 12.  ALLOWS USER TO SPECIFY JCG Ø35Ø
C                  SOME INTEGER PARAMETERS WHICH AFFECT THE METHOD.     JCG Ø36Ø
C          RPARM   REAL VECTOR OF LENGTH 12. ALLOWS USER TO SPECIFY SOME JCG Ø37Ø
C                  REAL PARAMETERS WHICH AFFECT THE METHOD.             JCG Ø38Ø
C          IER     OUTPUT INTEGER.  ERROR FLAG. (= IERR)                JCG Ø39Ø
C                                                                       JCG Ø4ØØ
C ... JCG SUBPROGRAM REFERENCES:                                        JCG Ø41Ø
C                                                                       JCG Ø42Ø
C          FROM ITPACK     BISRCH, CHGCON, DETERM, DFAULT, ECHALL,      JCG Ø43Ø
C                          ECHOUT, EIGVNS, EIGVSS, ITERM, ITICK,        JCG Ø44Ø
C                          ITJCG, ITOCK, IVFILL, PARCON, PERMAT,        JCG Ø45Ø
C                          PERROR, PERVEC, PJAC, PMULT, PRBNDX,         JCG Ø46Ø
C                          PSTOP, QSORT, SAXPY, SBELM, SCAL, SCOPY,     JCG Ø47Ø
C                          SDOT, SUM3, UNSCAL, VEVMW, VFILL, VOUT,      JCG Ø48Ø
C                          WEVMW                                        JCG Ø49Ø
C          SYSTEM          ABS, ALOG1Ø, AMAXØ, AMAX1, MOD, SQRT         JCG Ø5ØØ
C                                                                       JCG Ø51Ø
C     VERSION:  ITPACK 2C (MARCH 1982)                                  JCG Ø52Ø
C                                                                       JCG Ø53Ø
C     CODE WRITTEN BY:  DAVID KINCAID, ROGER GRIMES, JOHN RESPESS       JCG Ø54Ø
C                       CENTER FOR NUMERICAL ANALYSIS                   JCG Ø55Ø
C                       UNIVERSITY OF TEXAS                             JCG Ø56Ø
C                       AUSTIN, TX  78712                               JCG Ø57Ø
C                       (512) 471-1242                                  JCG Ø58Ø
C                                                                       JCG Ø59Ø
C     FOR ADDITIONAL DETAILS ON THE                                     JCG Ø6ØØ
C          (A) SUBROUTINE SEE TOMS ARTICLE 1982                         JCG Ø61Ø
C          (B) ALGORITHM  SEE CNA REPORT 15Ø                            JCG Ø62Ø
C                                                                       JCG Ø63Ø
C     BASED ON THEORY BY:  DAVID YOUNG, DAVID KINCAID, LOU HAGEMAN      JCG Ø64Ø
C                                                                       JCG Ø65Ø
C     REFERENCE THE BOOK:  APPLIED ITERATIVE METHODS                    JCG Ø66Ø
C                          L. HAGEMAN, D. YOUNG                         JCG Ø67Ø
C                          ACADEMIC PRESS, 1981                         JCG Ø68Ø
C                                                                       JCG Ø69Ø
```

587-P 1- 0

ALGORITHM 587
Two Algorithms for the Linearly Constrained Least Squares Problem

RICHARD J. HANSON and KAREN H. HASKELL
Sandia National Laboratories

Categories and Subject Descriptors: G.1.2. [**Numerical Analysis**]: Approximation—*least squares approximation*; G.1.6. [**Numerical Analysis**]: Optimization—*constrained optimization*; *least squares methods*; G.m [**Mathematics of Computing**]: Miscellaneous—*FORTRAN*

General Terms: Algorithms

Additional Key Words and Phrases: linear least squares solution, equality constraints, inequality constraints, nonnegativity constraints, inconsistent constraints, covariance matrix

1. INTRODUCTION

This paper discusses subroutines for computing numerical solutions of the following two linearly constrained linear least squares problems.

$$\begin{array}{lll} \textit{Problem NNLSE} & E\mathbf{x}=\mathbf{f} & \text{(equations to be exactly satisfied)} \\ & A\mathbf{x}\cong\mathbf{b} & \text{(equations to be approximately satisfied, least squares sense)} \\ & x_i\geq 0, & i=l+1,\ldots,n, \quad 0\leq l\leq n \end{array} \tag{1}$$

$$\begin{array}{lll} \textit{Problem LSEI} & E\mathbf{x}=\mathbf{f} & \text{(equations to be exactly satisfied)} \\ & A\mathbf{x}\cong\mathbf{b} & \text{(equations to be approximately satisfied, least squares sense)} \\ & G\mathbf{x}\geq\mathbf{h} & \text{(inequality constraints that the solution must satisfy)} \end{array} \tag{2}$$

In both problems the matrices E and A are real and of respective dimensions m_E by n and m_A by n. For Problem NNLSE, the variables $x_1, \ldots, x_l$ are free to have either sign. For Problem LSEI, the (real) inequality constraint matrix G is m_G by n. The right-side vectors **f**, **b**, and **h** that appear in the two problem statements have, respectively, m_E, m_A, and m_G components. The (unknown) solution vector **x** has n components.

While Problem LSEI of eq. (2) appears to be a more general problem than Problem NNLSE of eq. (1), it really is not. In fact, there are a number of ways to transform Problem LSEI into one of the forms of Problem NNLSE. Three ways of doing this are discussed in [3]. The method we have implemented is described on pages 101–102. The successful implementation of an algorithm for solving

Editing for this algorithm was supervised by Associate Editor W.J. Cody, Jr.
Received 9 February 1981; revised 19 March 1982; accepted 1 April 1982
Authors' addresses: R.J. Hanson, Numerical Mathematics Division 5642, Sandia National Laboratories, Albuquerque, NM 87185; K.H. Haskell, Applied Mathematics Division 2646, Sandia National Laboratories, Albuquerque, NM 87185.

Problem NNLSE is the key computational process. Nevertheless, it is important for applications such as constrained curve fitting [2] to have a subprogram that solves Problem LSEI of eq. (2) directly. We provide FORTRAN subprograms **WNNLS( )** and **LSEI( )** that solve the respective problems in eqs. (1) and (2).

In Section 2 we review mathematical and numerical analysis details pertinent to solving Problem LSEI. In Section 3 we review some necessary details for understanding our methods for solving Problem NNLSE. In Section 4 we summarize some features and advantages of the codes. These features include changing tolerances, scaling of data matrices, and optional computation of the covariance matrix. Section 5 presents a test subprogram **CLSTP( )**, which is included with the package. It solves the test problem with both subprograms. Section 6 contains installation guidelines and remarks.

2. SOLVING PROBLEM LSEI

In this section, we briefly review mathematical and algorithmic details needed to solve Problem LSEI of eq. (2) [3, pp. 101–102]. The overall process consists of four main parts.

Step 1 Problem LSEI is reduced to a subproblem with possibly fewer unknown variables and with all explicitly stated equality constraints removed.

Step 2 The problem resulting from step 1 is reduced to a new problem where the least squares matrix is a simple projection matrix and the right-side vector is zero.

Step 3 The problem resulting from step 2 is solved by reposing it as a dual problem. This dual problem consists of two special cases of Problem NNLSE, eq. (1).

Step 4 The solution obtained in step 3 is transformed to the solution of the original problem using translations, matrix multiplications, and the solution of triangular linear algebraic systems.

3. SOLVING PROBLEM NNLSE

The theoretical development for solving problem NNLSE of eq. (1) is presented in [3]. The fundamental point of this method involves a numerically stable implementation of a penalty function approach. The least squares equations are each weighted by a small parameter ϵ, chosen in the subprogram **WNNLS( )**. The augmented and weighted least squares system of eq. (3) is then solved.

$$\begin{bmatrix} E \\ \epsilon A \end{bmatrix} \mathbf{x} \equiv D \begin{bmatrix} E \\ A \end{bmatrix} \mathbf{x} \cong D \begin{bmatrix} \mathbf{f} \\ \mathbf{b} \end{bmatrix} \equiv \begin{bmatrix} \mathbf{f} \\ \epsilon \mathbf{b} \end{bmatrix}$$

$$D = \operatorname{diag}(\overbrace{1, \ldots, 1}^{m_E}, \overbrace{\epsilon, \ldots, \epsilon}^{m_A}) \tag{3}$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{y} \\ \mathbf{w} \end{bmatrix} \begin{matrix} \}l \\ \}n-l \end{matrix} \qquad \begin{matrix} \mathbf{y} \text{ unconstrained} \\ \mathbf{w} \geq \mathbf{0}. \end{matrix}$$

Part of the theoretical development in [3] shows that solutions of the weighted problem of eq. (3) converge to solutions of Problem NNLSE (if it is consistent) as $\epsilon \to 0$. Within the subprogram **WNNLS( )** eq. (3) is solved only once with a value of ϵ that is chosen to achieve full working accuracy in the solution. The value used in **WNNLS( )** is defined by

$$\epsilon^2 = \frac{10^{-4}\eta}{\gamma}, \tag{4}$$

where $\gamma = \| \begin{smallmatrix} E \\ A \end{smallmatrix} \|$, ($\| \cdot \|$ = subordinate matrix norm of l_∞ vector norm), and η = machine relative arithmetic precision.

The algorithm for solving eq. (3) with ϵ as defined in eq. (4) proceeds in two main steps. First we compute a (minimum-length) solution for the unconstrained

variables in terms of the constrained variables. Solving for the unconstrained variables is primarily a triangularization operation. In the second main step of the process we solve for the constrained variables. This is an iterative process, that is, it is Algorithm NNLS of [7, Chap. 23]. Certain crucial differences in numerical tests are needed because of the penalty parameter ϵ that multiplies the least squares equations. These tests are discussed in [3].

4. USAGE SUGGESTIONS AND SUBPROGRAM OPTIONS

In Sections 2 and 3 we have outlined solution methods for solving Problem LSEI of eq. (2) and Problem NNLSE of eq. (1). As shown in [3], computing the solution of Problem NNLSE can be regarded as the core computation in solving constrained linear least squares problems.

The most satisfactory method from the standpoint of accuracy and stability is to introduce slack variables into the inequality constraints of Problem LSEI [3]. This problem is then solved using subprogram **WNNLS( )**. The results of solving a bounded variable Hilbert matrix problem summarized in [3] suggest that subprogram **WNNLS( )** continues to compute acceptable solutions even as the problems become increasingly ill-conditioned.

The use of subprogram **WNNLS( )** with the slack variable formulation does have a disadvantage compared to subprogram **LSEI( )**. For most problems, **WNNLS( )** will require more computing time and storage than **LSEI( )**. This is due to the larger number of problem variables in the slack variable formulation. The advantage of efficiency with **LSEI( )** may be countered by the simultaneous occurrence of poor conditioning and rounding errors. (This can occur with a poorly conditioned least squares problem.) Owing to the poor conditioning and rounding error, the feasible constraint region can be mapped to one that is infeasible. Instances of this are shown in the results of solving the bounded variable Hilbert matrix problem summarized in [3].

The choice between the two subprograms is a time and storage versus stability trade-off. Specifically, in the case of a poorly conditioned least squares problem, **WNNLS( )** might obtain a solution when **LSEI( )** cannot. As illustrated in [3], subprogram **WNNLS( )** can also be used to extend the notion of solution for problems with infeasible constraints.

Occasionally, a user of subprogram **LSEI( )** will need the covariance matrix of the least squares solution variables of minimum length. This is returned as an output matrix if the user wants it. It is an unbiased estimate of the covariance matrix for the minimum-length solution of an equality constrained least squares problem *with no inequalities.* This is developed in [4] and [6].

When inequalities are included, certain additional mathematical problems must be considered. These have to do with the behavior of the set of inequalities chosen by the algorithm to be equalities. The question is as follows: What is the sensitivity of these equalities as the data are allowed to vary within its uncertainty? Inequalities may move from being satisfied as equalities to strict inequalities as the data are perturbed. The covariance matrix computed by **LSEI( )** is based on the assumption that the set of equalities does not change when the solution is perturbed. No comprehensive theory is known to the authors for determining the matrix when the set of equalities does change. The user must keep these facts in mind when interpreting the covariance matrix for Problem LSEI with inequalities.

The remainder of this section describes parameters within **LSEI( )** and **WNNLS( )** which can optionally be changed by the user. These options fall into the three following groups.

(A) Computation of the covariance matrix.
(B) Column scaling of the data matrix.
(C) Redefinition of tolerances used for determining ranks of problem matrices.

Changes to any number of these parameters can be specified as the linked-list input in the array **PRGOPT(*)**. Precise instructions for defining **PRGOPT(*)** are found in the usage prologues for **LSEI( )** and **WNNLS( )**. If the user is satisfied with the nominal subprogram features, it is only necessary to set **PRGOPT(1)=1.**

Remarks about A: Nominally the covariance matrix is not computed by **LSEI( )**.

Remarks about B: Column scaling of the form $\mathbf{x} = D\mathbf{y}$ is always performed by **LSEI( )** and **WNNLS( )**. Nominally D is the identity matrix. Another option here is a choice for D such that each nonzero column of the entire scaled data matrix has length one. The user can also specify an arbitrary D.

Remarks about C: The user can change tolerances t_E and t_A in **LSEI( )** and tolerance t_W in **WNNLS( )**. The nominal values of t_E, t_A, and t_W are $\eta^{1/2}$, where η is the relative arithmetic precision of the machine. The parameter t_E is used in approximating the rank of the equality constraint matrix E of eq. (2). Its role is discussed near the end of [3, Sec. 1].

The parameter t_A is used in approximating the rank of the least squares matrix that results from eliminating the equality constraints from eq. (2). It is used to compute the factor τ, which is t_A times the norm of this reduced least squares matrix. Then τ is used in Algorithm HFTI [7, Chap. 14].

The parameter t_W is used by **WNNLS( )** to compute the rank of the row-scaled least squares matrix as discussed in [3, Sec. 3.1].

5. REMARKS ON THE TESTING SUBPROGRAM CLSTP( )

The subprogram **CLSTP (KLOG, COND, ISTAT)** constructs and solves a constrained least squares problem that has a known solution and known condition numbers [7, Chap. 9]. The problem generated is stated in eq. (2). The matrices A, E, and G are computed using formulas

$$A = U_1 S_1 V_1^{\mathrm{T}}$$

$$E = U_2 S_2 V_2^{\mathrm{T}}$$

and

$$G = U_3 S_3 V_3^{\mathrm{T}}.$$

The problem dimensions are specified by using five integer parameters k_A, k_E, k_G, k_I, and k_n to compute $m_A = 2^{k_A}$, $m_E = 2^{k_E}$, $m_G = 2^{k_G}$, and $n = 2^{k_n}$. The integer $m_I = 2^{k_I}$ denotes the number of inequality constraints that are to be satisfied as strict inequalities. These five integers are passed to **CLSTP( )** in the array **KLOG(*)** in the order indicated. If any of the values k_A, k_E, k_G, k_I, or k_n are less than zero, the respective values m_A, m_E, m_G, m_I, or m_n are set to zero. No computation is performed if $n = 0$.

Arrays within **CLSTP( )** currently have fixed dimensions that require k_A, k_E, k_G, k_I, and k_n to all be less than or equal to 5. Instructions for increasing the array dimensions are given as comments within **CLSTP( )**.

The matrices U_J and V_J are symmetric orthogonal Hadamard matrices of dimension $n = 2^k$ generated by the recursion

$n := 1$

$U := 1$

For $l = 1, \ldots, k$

$$U := \begin{bmatrix} U & : & U \\ U & : & -U \end{bmatrix}$$

$n := n + n$

End For

$$U := n^{-1/2}U$$

The matrices S_J, $J = 1, 2, 3$, are rectangular diagonal matrices. The extreme diagonal terms are κ_i and 1, where $\kappa_J \equiv$ **COND(J)**. The intermediate diagonal terms are generated in the open interval $(1, \kappa_J)$ using the random number generator **RAN( )**. The output value of $t =$ **RAN(ISEED)** satisfies $0 < t < 1$. The intermediate diagonal terms are successively computed as $1 + t(\kappa_J - 1)$. Initially, **ISEED** is set to 100001 in **CLSTP( )**.

The n-vector $\hat{\mathbf{x}} = (1, \ldots, 1)^T$ is used to generate the vectors

$$\mathbf{f} = E\hat{\mathbf{x}}$$

$$\hat{\mathbf{b}} = A\hat{\mathbf{x}}$$

and

$$\hat{\mathbf{h}} = G\hat{\mathbf{x}}.$$

We add a vector $\tilde{\mathbf{b}}$ to $\hat{\mathbf{b}}$ that is orthogonal to the column space of A. This is given by

$$\tilde{\mathbf{b}} = U_1(0, \ldots, 0, g_{n+1}, \ldots, g_{m_A}),$$

where

$$g_i = \textbf{RAN(ISEED)} \cdot \| \hat{\mathbf{b}} \| \cdot \sigma, \qquad i = n + 1, \ldots, m_A.$$

The value of σ is specified by the variable **ANSR** in **CLSTP( )**. It is currently set to 0.01. The right-side vector for the least squares equations in eq. (2) is $\mathbf{b} \equiv \hat{\mathbf{b}} + \tilde{\mathbf{b}}$.

The right-side vector for the inequality constraints is constructed by making the first m_I constraints strict inequalities. This is done by defining the right-side vector as $\mathbf{h} = \hat{\mathbf{h}} - \tilde{\mathbf{h}}$, where

$$\tilde{\mathbf{h}} = (h_1, \ldots, h_{m_I}, 0, \ldots, 0^T),$$

$$h_i = \textbf{RAN(ISEED)} \cdot \| \hat{\mathbf{h}} \|, \qquad i = 1, \ldots, m_I.$$

These techniques for generating problems with known solutions are similar to those discussed in [9, pp. 6–9]. One might obtain different sets of test problems on machines with differing arithmetic characteristics. Part of this is due to a different sequence of numbers generated by **RAN( )**.

We have found that column scaling is sometimes required for solving eqs. (1) and (2). In particular, when using 32-bit floating-point arithmetic, problems generated by **CLSTP( )** using the published test data occasionally failed to pass the tests when no column scaling was done. Thus the option array input for calls to both **LSEI( )** and **WNNLS( )** are set so that unit length column scaling is performed on all the tests.

After subprogram **LSEI( )** has computed an approximate solution $\mathbf{x}'$ for this particular form of eq. (2), and subprogram **WNNLS( )** has solved for an approximate solution $\mathbf{x}''$ of the system

$$E\mathbf{x} = \mathbf{f}$$

$$A\mathbf{x} \cong \mathbf{b}$$

$$G\mathbf{x} - \mathbf{h} = \mathbf{w}$$

for the unknown $(\mathbf{x}^T, \mathbf{w}^T)^T$, we compute the differences $\mathbf{dx}_1 = \mathbf{x}' - \hat{\mathbf{x}}$ and $\mathbf{dx}_2 = \mathbf{x}'' - \hat{\mathbf{x}}$. A test is made on the value of $\| \mathbf{dx}_i \|$ to ensure that $\mathbf{x}'$ or $\mathbf{x}''$ is as accurate as it deserves to be. The test of the subprogram has failed if the corresponding $\| \mathbf{dx}_i \|$ is too large. Otherwise the test has passed and $\mathbf{x}'$ or $\mathbf{x}''$ is an acceptable approximation of $\mathbf{x}$. With

$$\rho = \| \tilde{\mathbf{b}} \| / \| \hat{\mathbf{b}} \|$$

$$\kappa = \kappa_1 = \text{condition number of } A$$

η = relative arithmetic precision

$\mu = \max(m_A, n)$

$\nu = \min(m_A, n)$

$\phi = 100$

each test has passed if and only if

$$\frac{\| \mathbf{dx}_i \|}{\| \hat{\mathbf{x}} \|} \leq \kappa(1 + \kappa\rho)\eta[(6\mu - 3\nu)\nu]\phi.$$

The output value of **ISTAT** is set as follows:

ISTAT = 1 means both **LSEI( )** and **WNNLS( )** failed.
= 2 means **WNNLS( )** passed but **LSEI( )** failed.
= 3 means **LSEI( )** passed but **WNNLS( )** failed.
= 4 means both **LSEI( )** and **WNNLS( )** passed.

This measure for $\| \mathbf{dx}_i \|$ is based on combining the estimate for the norm of the matrix H of the nearby problem that $\mathbf{x}'$ solves (without constraints), $(A + H)\mathbf{x}' \cong \mathbf{b}$, [7, Chap. 13], together with the perturbation bounds of [7, Chap. 9].

It may be necessary to increase the value of ϕ slightly on some machines.

A short main program, **CLSTST**, is provided with the algorithm. Also provided are 11 data cards that are read by **CLSTST** from FORTRAN unit = 5. Each pair of the first 10 cards specifies a distinct test case. The last (eleventh) card terminates the program execution.

The subprogram **CLSTP( )** prints the computed values of the least squares residual vector length and the vectors $\mathbf{dx}_i$ for both **WNNLS( )** and **LSEI( )**. Also printed in **CLSTP( )** are the computed ranks of the equality constraint and reduced least squares matrices returned by **LSEI( )**. The arrays **KLOG(I), I = 1 to 5**, and **COND(I), J = 1 to 3**, and the value of **ISTAT** returned from **CLSTP( )** are printed by **CLSTST**. Printing is done on FORTRAN unit = 6.

6. INSTALLATION GUIDELINES AND REMARKS

This section contains information for installing subprograms **LSEI( )** and **WNNLS( )**.

Included in the package are seven groups of subprograms.

(1) **LSEI, LSI, LPDP**
(2) **WNNLS, WNLSM, WNLIT**
(3) **HFTI, H12, DIFF** from [7]
(4) **SDOT, SSCAL, SASUM, SAXPY, SNRM2, SCOPY, SSWAP, ISAMAX, SROTM, SROTMG** from [8]. (For double-precision usage **DDOT, DSCAL, DASUM, DAXPY, DNRM2, DCOPY, DSWAP, IDAMAX, DROTM, DROTMG**.)
(5) **XERROR, XERRWV, XERABT, XERCLR, XERCTL, XERDMP, XERMAX, XERPRT, XERSAV, XGETF, XGETUA, XGETUN, XSETF, XSETUA, XSETUN, FDUMP, J4SAVE, S88FMT, NUMXER** from [5]. (The subprogram **NUMXER** is included for completeness but is not used in this package.)
(6) **I1MACH** based on [1]
(7) **CLSTST, CLSTP, RAN** (test package)

All of the subprograms are written in 1966 American National Standard portable FORTRAN. The only machine-sensitive subprogram is **I1MACH( )**. It provides two environmental parameters required by the error-handling subprograms **XERROR( )** and **XERRWV( )**. This will require modification of **I1MACH( )** at each host site. FORTRAN **DATA** statements defining the values of all the required constants are available for many machines in comments within the subprogram. The appropriate set of commented statements must be activated. If the values for your machine are not there, they should be provided in the order

corresponding to the description near the beginning of **I1MACH( )**. Machines for which these constants are provided are Honeywell 600/6000, IBM 360/370, Xerox Sigma, CDC 6000/7000, PDP-10 (KA and KI processors), PDP-11 (16- and 32-bit arithmetic), Burroughs 5700/6700/7700, UNIVAC 1100, Data General Eclipse, Harris, VAX, and CRAY. In addition, the user must open or declare the FORTRAN unit, designated in **I1MACH(4)**, where any error messages will be written.

We strongly recommend that calls to the error-handling subprograms **XERROR( )** and **XERRWV( )** be left intact. If the size or complexity of the error-handling package presents a problem on a particular machine, we suggest that the subprograms **XERROR( )** and **XERRWV( )** be replaced by shorter, machine-sensitive versions. These replacements should, minimally, print the character string comprising the error message and the specified data values. Usage of the full error-handling package is discussed in [5].

To convert the package for double-precision usage, follow the editing instructions at the beginning of each subprogram in groups 1, 2, 3, and 7 above. Use the double-precision version of the BLAS in group 4. No conversion is required for subprograms in groups 5 and 6.

ACKNOWLEDGMENTS

We thank two anonymous referees for providing suggestions that clarified the presentation of this algorithm.

REFERENCES

1. Fox, P.A., Hull, A.D., and Schryer, N.D. The PORT mathematical subroutine library. *ACM Trans. Math. Softw. 4*, 2 (June 1978), 104–126.
2. Hanson, R.J. *Constrained Least Squares Curve Fitting to Discrete Data Using B-splines—A User's Guide.* Available as Sandia National Laboratories Tech. Rep. SAND78-1291, Sandia National Laboratories, Albuquerque, N.Mex., Feb. 1979.
3. Haskell, K.H., and Hanson, R.J. An algorithm for linear least squares problems with equality and nonnegativity constraints. *Math. Program. 21* (1981), 98–118.
4. Haskell, K.H., and Hanson, R.J. Selected algorithms for the linearly constrained least squares problem—A user's guide. Tech. Rep. SAND78-1290, Sandia National Laboratories, Albuquerque, N.Mex., 1979.
5. Jones, R.E. SLATEC common mathematical library error handling package. Tech. Rep. SAND78-1189, Sandia National Laboratories, Albuquerque, N.Mex., 1978.
6. Lawson, C.L. *The Covariance Matrix for the Solution Vector of an Equality-Constrained Least-Squares Problem.* Available as Jet Propulsion Laboratories Tech. Memo 33-807, Jet Propulsion Laboratories, Pasadena, Calif., Dec. 1976.
7. Lawson, C.L., and Hanson, R.J. *Solving Least Squares Problems.* Prentice-Hall, Englewood Cliffs, N.J., 1974.
8. Lawson, C.L., Hanson, R.J., Kincaid, D.R., and Krogh, F.T. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Softw. 5*, 3 (Sept. 1979), 308–323.
9. Wampler, R.H. Problems used in testing the efficiency and accuracy of the modified Gram-Schmidt least squares algorithm. NBS Tech. Note 1126, Aug. 1980.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service .]

```
      SUBROUTINE LSEI(W, MDW, ME, MA, MG, N, PRGOPT, X, RNORME, RNORML, LSEI  1Ø
     * MODE, WS, IP)                                                    LSEI  2Ø
C                                                                       LSEI  3Ø
C     DIMENSION W(MDW,N+1),PRGOPT(*),X(N),                              LSEI  4Ø
C     WS(2*(ME+N)+K+(MG+2)*(N+7)),IP(MG+2*N+2)                          LSEI  5Ø
C     ABOVE, K=MAX(MA+MG,N).                                            LSEI  6Ø
C                                                                       LSEI  7Ø
C     ABSTRACT                                                          LSEI  8Ø
C                                                                       LSEI  9Ø
C     THIS SUBPROGRAM SOLVES A LINEARLY CONSTRAINED LEAST SQUARES       LSEI 1ØØ
C     PROBLEM WITH BOTH EQUALITY AND INEQUALITY CONSTRAINTS, AND, IF THELSEI 11Ø
```

```
C     USER REQUESTS, OBTAINS A COVARIANCE MATRIX OF THE SOLUTION        LSEI 12Ø
C     PARAMETERS.                                                       LSEI 13Ø
C                                                                       LSEI 14Ø
C     SUPPOSE THERE ARE GIVEN MATRICES E, A AND G OF RESPECTIVE         LSEI 15Ø
C     DIMENSIONS ME BY N, MA BY N AND MG BY N, AND VECTORS F, B AND H OFLSEI 16Ø
C     RESPECTIVE LENGTHS ME, MA AND MG.  THIS SUBROUTINE SOLVES THE     LSEI 17Ø
C     LINEARLY CONSTRAINED LEAST SQUARES PROBLEM                        LSEI 18Ø
C                                                                       LSEI 19Ø
C                 EX = F, (E ME BY N) (EQUATIONS TO BE EXACTLY          LSEI 2ØØ
C                                      SATISFIED)                       LSEI 21Ø
C                 AX = B, (A MA BY N) (EQUATIONS TO BE                  LSEI 22Ø
C                                      APPROXIMATELY SATISFIED,         LSEI 23Ø
C                                      LEAST SQUARES SENSE)             LSEI 24Ø
C                 GX.GE.H,(G MG BY N) (INEQUALITY CONSTRAINTS)          LSEI 25Ø
C                                                                       LSEI 26Ø
C     THE INEQUALITIES GX.GE.H MEAN THAT EVERY COMPONENT OF THE PRODUCT LSEI 27Ø
C     GX MUST BE .GE. THE CORRESPONDING COMPONENT OF H.                 LSEI 28Ø
C                                                                       LSEI 29Ø
C     IN CASE THE EQUALITY CONSTRAINTS CANNOT BE SATISFIED, A           LSEI 3ØØ
C     GENERALIZED INVERSE SOLUTION RESIDUAL VECTOR LENGTH IS OBTAINED   LSEI 31Ø
C     FOR F-EX. THIS IS THE MINIMAL LENGTH POSSIBLE FOR F-EX.           LSEI 32Ø
C                                                                       LSEI 33Ø
C                                                                       LSEI 34Ø
C     ANY VALUES ME.GE.Ø, MA.GE.Ø, OR MG.GE.Ø ARE PERMITTED.  THE       LSEI 35Ø
C     RANK OF THE MATRIX E IS ESTIMATED DURING THE COMPUTATION. WE CALL LSEI 36Ø
C     THIS VALUE KRANKE. IT IS AN OUTPUT PARAMETER IN IP(1) DEFINED     LSEI 37Ø
C     BELOW. USING A GENERALIZED INVERSE SOLUTION OF EX=F, A REDUCED    LSEI 38Ø
C     LEAST SQUARES PROBLEM WITH INEQUALITY CONSTRAINTS IS OBTAINED.    LSEI 39Ø
C     THE TOLERANCES USED IN THESE TESTS FOR DETERMINING THE RANK       LSEI 4ØØ
C     OF E AND THE RANK OF THE REDUCED LEAST SQUARES PROBLEM ARE        LSEI 41Ø
C     GIVEN IN SANDIA TECH. REPT. SAND 78-129Ø. THEY CAN BE             LSEI 42Ø
C     MODIFIED BY THE USER IF NEW VALUES ARE PROVIDED IN                LSEI 43Ø
C     THE OPTION LIST OF THE ARRAY PRGOPT(*).                           LSEI 44Ø
C                                                                       LSEI 45Ø
C     THE EDITING REQUIRED TO CONVERT THIS SUBROUTINE FROM SINGLE TO    LSEI 46Ø
C     DOUBLE PRECISION INVOLVES THE FOLLOWING CHARACTER STRING CHANGES. LSEI 47Ø
C     USE AN EDITING COMMAND (CHANGE) /STRING-1/(TO)STRING-2/.          LSEI 48Ø
C     (START EDITING AT LINE WITH C++ IN COLS. 1-3.)                    LSEI 49Ø
C     /REAL (12 BLANKS)/DOUBLE PRECISION/,/SASUM/DASUM/,/SDOT/DDOT/,    LSEI 5ØØ
C     /SNRM2/DNRM2/,/ SQRT/ DSQRT/,/ ABS/ DABS/,/AMAX1/DMAX1/,          LSEI 51Ø
C     /SCOPY/DCOPY/,/SSCAL/DSCAL/,/SAXPY/DAXPY/,/SSWAP/DSWAP/,/EØ/DØ/,  LSEI 52Ø
C     /, DUMMY/,SNGL(DUMMY)/,/SRELPR/DRELPR/                            LSEI 53Ø
C                                                                       LSEI 54Ø
C     WRITTEN BY R. J. HANSON AND K. H. HASKELL.  FOR FURTHER MATH.     LSEI 55Ø
C     AND ALGORITHMIC DETAILS SEE SANDIA LABORATORIES TECH. REPTS.      LSEI 56Ø
C     SAND 77-Ø552, (1978), SAND 78-129Ø, (1979), AND                   LSEI 57Ø
C     MATH. PROGRAMMING, VOL. 21, (1981), P.98-118.                     LSEI 58Ø


      SUBROUTINE WNNLS(W, MDW, ME, MA, N, L, PRGOPT, X, RNORM, MODE,    WNN   1Ø
     * IWORK, WORK)                                                     WNN   2Ø
C                                                                       WNN   3Ø
C     DIMENSION W(MDW,N+1),PRGOPT(*),X(N),IWORK(M+N),WORK(M+5*N)        WNN   4Ø
C                                                                       WNN   5Ø
C     ABSTRACT                                                          WNN   6Ø
C                                                                       WNN   7Ø
C     THIS SUBPROGRAM SOLVES A LINEARLY CONSTRAINED LEAST SQUARES       WNN   8Ø
C     PROBLEM.  SUPPOSE THERE ARE GIVEN MATRICES E AND A OF             WNN   9Ø
C     RESPECTIVE DIMENSIONS ME BY N AND MA BY N, AND VECTORS F          WNN  1ØØ
C     AND B OF RESPECTIVE LENGTHS ME AND MA.  THIS SUBROUTINE           WNN  11Ø
C     SOLVES THE PROBLEM                                                WNN  12Ø
C                                                                       WNN  13Ø
C               EX = F, (EQUATIONS TO BE EXACTLY SATISFIED)             WNN  14Ø
C                                                                       WNN  15Ø
C               AX = B, (EQUATIONS TO BE APPROXIMATELY SATISFIED,       WNN  16Ø
C                        IN THE LEAST SQUARES SENSE)                    WNN  17Ø
C                                                                       WNN  18Ø
C               SUBJECT TO COMPONENTS L+1,...,N NONNEGATIVE             WNN  19Ø
C                                                                       WNN  2ØØ
C     ANY VALUES ME.GE.Ø, MA.GE.Ø AND Ø.LE. L .LE.N ARE PERMITTED.      WNN  21Ø
C                                                                       WNN  22Ø
C     THE PROBLEM IS REPOSED AS PROBLEM WNNLS                           WNN  23Ø
C                                                                       WNN  24Ø
```

```
C               (WT*E)X = (WT*F)                                        WNN  25Ø
C               (   A)    (   B), (LEAST SQUARES)                       WNN  26Ø
C               SUBJECT TO COMPONENTS L+1,...,N NONNEGATIVE.            WNN  27Ø
C                                                                       WNN  28Ø
C     THE SUBPROGRAM CHOOSES THE HEAVY WEIGHT (OR PENALTY PARAMETER) WT.WNN  29Ø
C                                                                       WNN  3ØØ
C     THE PARAMETERS FOR WNNLS ARE                                      WNN  31Ø
C                                                                       WNN  32Ø
C     INPUT..                                                           WNN  33Ø
C                                                                       WNN  34Ø
C     W(*,*),MDW,  THE ARRAY W(*,*) IS DOUBLE SUBSCRIPTED WITH FIRST    WNN  35Ø
C     ME,MA,N,L    DIMENSIONING PARAMETER EQUAL TO MDW.  FOR THIS       WNN  36Ø
C                  DISCUSSION LET US CALL M = ME + MA.  THEN MDW        WNN  37Ø
C                  MUST SATISFY MDW.GE.M.  THE CONDITION MDW.LT.M       WNN  38Ø
C                  IS AN ERROR.                                         WNN  39Ø
C                                                                       WNN  4ØØ
C                  THE ARRAY W(*,*) CONTAINS THE MATRICES AND VECTORS   WNN  41Ø
C                                                                       WNN  42Ø
C                       (E  F)                                          WNN  43Ø
C                       (A  B)                                          WNN  44Ø
C                                                                       WNN  45Ø
C                  IN ROWS AND COLUMNS 1,...,M AND 1,...,N+1            WNN  46Ø
C                  RESPECTIVELY.  COLUMNS 1,...,L CORRESPOND TO         WNN  47Ø
C                  UNCONSTRAINED VARIABLES X(1),...,X(L).  THE          WNN  48Ø
C                  REMAINING VARIABLES ARE CONSTRAINED TO BE            WNN  49Ø
C                  NONNEGATIVE.  THE CONDITION L.LT.Ø .OR. L.GT.N IS    WNN  5ØØ
C                  AN ERROR.                                            WNN  51Ø
C                                                                       WNN  52Ø
C     PRGOPT(*)    THIS ARRAY IS THE OPTION VECTOR.                     WNN  53Ø
C                  IF THE USER IS SATISFIED WITH THE NOMINAL            WNN  54Ø
C                  SUBPROGRAM FEATURES SET                              WNN  55Ø
C                                                                       WNN  56Ø
C                  PRGOPT(1)=1 (OR PRGOPT(1)=1.Ø)                       WNN  57Ø
```

ALGORITHM 588
Fast Hankel Transforms Using Related and Lagged Convolutions

WALTER L. ANDERSON
U.S. Geological Survey

Categories and Subject Descriptors: F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*computation of transforms*; G.1.2 [**Numerical Analysis**]: Approximation—*linear approximation*; G.m [**Mathematics of Computing**]: Miscellaneous—*FORTRAN*

General Terms: Algorithms

Additional Key Words and Phrases: Hankel transforms of integer order, Bessel functions of the first kind, convolution integrals, linear digital filters

DESCRIPTION

Subprogram **HANKEL( )**, for evaluating complex Hankel transforms, is a complement of [1], where the algorithm is discussed in more detail. A double-precision real version, subprogram **DHANKL( )**, is also provided. The calling sequence for **DHANKL( )** is ordered the same way as is described below for **HANKEL( )**, except all **REAL** and **COMPLEX** declarations in **HANKEL( )** are declared **DOUBLE PRECISION** in **DHANKL( )**.

REFERENCES

1. ANDERSON, W.L. Fast Hankel transforms using related and lagged convolutions. *ACM Trans. Math. Softw. 8*, 4 (Dec. 1982), 344–368.

ALGORITHM

[Summary information and a part of the listing is printed here. The entire package, including subprogram **HANKEL( )** and three drivers, followed by **DHANKL( )** and one driver, is available from the ACM Algorithms Distribution Service.]

Module Names

SUBROUTINE HANKEL( )
[MAIN PROGRAM—HANKEL TEST 1]
COMPLEX FUNCTION FUNT1( )
FUNCTION EXPM( )
[MAIN PROGRAM—HANKEL TEST 2]
COMPLEX FUNCTION FUNT2( )
FUNCTION SINHM( )
[MAIN PROGRAM—HANKEL TEST 3]
COMPLEX FUNCTION FUNT3( )

SUBROUTINE DHANKL( )
[MAIN PROGRAM—DHANKL TEST 1D]

Received 7 March 1980; revised 5 May 1982; accepted 29 July 1982
Author's address: U.S. Geological Survey, Box 25046, Mail Stop 964, Denver Federal Center, Denver, CO 80225.

DOUBLE PRECISION FUNCTION DFUNT1( )
DOUBLE PRECISION FUNCTION DEXPM( )

```
      SUBROUTINE HANKEL(BMAX,NB,NREL,TOL,NTOL,NORD,FUN1,IJREL,ZWORK,     HAN   1Ø
     * ZANS,ARG,NOFUN1,IERR)                                             HAN   2Ø
C=====================================================================HAN   3Ø
      INTEGER NB,NREL,NTOL,NORD(NREL),IJREL(2,NREL),NOFUN1,IERR          HAN   4Ø
      REAL BMAX,TOL,ARG(NB)                                              HAN   5Ø
      COMPLEX ZWORK(283,NREL),ZANS(NB,NREL)                              HAN   6Ø
C=====================================================================HAN   7Ø
C                                                                        HAN   8Ø
C  PURPOSE                                                               HAN   9Ø
C                                                                        HAN  1ØØ
C     THE PURPOSE OF SUBPROGRAM HANKEL IS TO PROVIDE IN SINGLE PRECISIONHAN  11Ø
C    A GENERAL ALGORITHM FOR FAST COMPLEX HANKEL TRANSFORMS OF ORDERS    HAN  12Ø
C    Ø AND 1 USING RELATED AND LAGGED CONVOLUTIONS.                      HAN  13Ø
C                                                                        HAN  14Ø
C  AUTHOR                                                                HAN  15Ø
C                                                                        HAN  16Ø
C    ANDERSON, W.L., U.S. GEOLOGICAL SURVEY, DENVER, COLORADO.           HAN  17Ø
C                                                                        HAN  18Ø
C  REFERENCES                                                            HAN  19Ø
C                                                                        HAN  2ØØ
C    1. ANDERSON, W.L., IMPROVED DIGITAL FILTERS FOR EVALUATING          HAN  21Ø
C       FOURIER AND HANKEL TRANSFORM INTEGRALS. N.T.I.S REPT.            HAN  22Ø
C       PB-242-8ØØ, SPRINGFIELD, VA., 1975.                              HAN  23Ø
C                                                                        HAN  24Ø
C    2. ANDERSON, W.L., NUMERICAL INTEGRATION OF RELATED HANKEL          HAN  25Ø
C       TRANSFORMS OF ORDERS Ø AND 1 BY ADAPTIVE DIGITAL FILTERING.      HAN  26Ø
C       GEOPHYSICS 44 (JULY 1979), 1287-13Ø5.                            HAN  27Ø
C                                                                        HAN  28Ø
C  LANGUAGE                                                              HAN  29Ø
C                                                                        HAN  3ØØ
C    ANS-FORTRAN (X3.9-1966) IS USED, WITH THE EXCEPTION OF THE          HAN  31Ø
C    CHARACTERS F,!,&,:,1,J APPEARING IN SOME COMMENT STATEMENTS.        HAN  32Ø
```

ALGORITHM 589
SICEDR: A FORTRAN Subroutine for Improving the Accuracy of Computed Matrix Eigenvalues

JACK J. DONGARRA
Argonne National Laboratory

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*eigenvalues*; G.m [**Mathematics of Computing**]: Miscellaneous—*FORTRAN*
General Terms: Algorithms
Additional Key Words and Phrases: Matrix eigensystems, iterative method; eigensystem improvement

1. INTRODUCTION

SICEDR is a FORTRAN subroutine for improving the accuracy of a computed real eigenvalue and improving or computing the associated eigenvector. It is first used to generate information during the determination of the eigenvalues by the Schur decomposition technique. In particular, the Schur decomposition technique results in an orthogonal matrix Q and an upper quasi-triangular matrix T, such that

$$A = QTQ^{\mathrm{T}}. \tag{1.1}$$

Matrices A, Q, and T and the approximate eigenvalue, say λ, are then used in the improvement phase. **SICEDR** uses an iterative method similar to iterative improvement for linear systems to improve the accuracy of λ and improve or compute the eigenvector x in $O(n^2)$ work, where n is the order of the matrix A. The method used in **SICEDR** is described in [1, 5].

2. USAGE

For a description of the calling sequence, see the listing presented at the end of this paper.

SICEDR factors the matrix into its Schur decomposition, and this is termed the pre-**SICE** phase. A modification of the EISPACK routine **HQR2** is used for this purpose (see [4, pp. 100–101] for details).

During the improvement phase (or **SICE** phase), **SICEDR** is called with matrices A, T, and Q and the approximate eigenvalue W. On return from **SICEDR** the improved eigenvalue is stored in W and the improved or computed eigenvector in X. W and X contain the corrected eigenpair produced by the method at the next to last iteration. A still more accurate eigenpair can be formed if on return from **SICEDR** the user adds W to CW and X to CX in double

Received 15 October 1980; revised 14 July 1982; accepted 20 July 1982
This work was supported in part by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under Contract W-31-109-Eng-38 and in part at the University of New Mexico by the Computer Science Section of the National Science Foundation under Contract MCS 7603297.
Author's address: Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439.

precision and saves the results in double precision, where CW and CX contain the corrections to W and X, respectively, at the final iteration when the method terminates.

The routine **SICEDR** is provided as a driver to simplify usage. **SICEDR** performs both the pre-**SICE** phase (reduction to upper form) and the **SICE** phase (improvement).

In addition to the EISPACK package, the LINPACK [2] routine **STRSL** and the BLAS [3] package are used to perform fundamental operations.

To produce a double-precision version of **SICEDR** in addition to making the obvious changes, such as changing variables to double precision and using double-precision intrinsic functions, it is necessary to replace routine **STRSL** by **DTRSL** and use the double-precision version of the BLAS. In addition, there are two critical parts of the calculation which *must* be performed in an extended precision. In the single-precision version, the routine **SDDDOT** and **SDADD** perform double-precision accumulations. In a double-precision version, they must be replaced by, say **DQQDOT** and **DQADD**, the extended precision counterparts. These extended precision routines would accumulate in quadruple precision, a nonstandard FORTRAN construct. Such routines are not included here, but can be constructed from the BLAS routines **DQDOTI** and **DQDOTA.**

3. SUMMARY OF THE ALGORITHM

We begin with a brief discussion of the basic algorithm as described in [1].

If λ, x is an approximate eigenpair, and $\lambda + \mu$, $x + \tilde{y}$ is a neighboring eigenpair, then

$$A(x + \tilde{y}) = (\lambda + \mu)(x + \tilde{y}), \tag{3.1}$$

this relation being exact. We assume that x is normalized so that $\| x \|_\infty = 1 = x_s$, and we remove the degree of arbitrariness in $\tilde{y}$ by requiring that $\tilde{y}_s = 0$. Rearranging this equation, we have

$$(A - \lambda I)\tilde{y} - \mu x = \lambda x - Ax + \mu\tilde{y}, \tag{3.2}$$

where the last term on the right will be of second order in the errors of λ, x. The equation above may be simplified by the introduction of a vector y defined by

$$y^{\mathrm{T}} = (y_1, y_2, \ldots, y_{s-1}, \mu, y_{s+1}, \ldots, y_n), \tag{3.3}$$

so that y gives the full information on both μ and $\tilde{y}$. The above equation then becomes

$$By = r + y_s\tilde{y}, \tag{3.4}$$

where $r = \lambda x - Ax$ is the residual vector corresponding to λ, x and B is the matrix $A - \lambda I$ with column s replaced by $-x$.

When the original approximate eigenpairs have been found by Francis' double QR algorithm, we have an orthogonal matrix Q and a quasi upper triangular matrix T, such that

$$A = QTQ^{\mathrm{T}}. \tag{3.5}$$

T is triangular apart from possible 2×2 diagonal blocks corresponding to complex conjugate eigenvalues. We solved a succession of linear systems of the form (3.4) with varying right-hand sides g. It will be convenient to introduce the generic notation $Z - \lambda I = Z_\lambda$ and $(Z - \lambda I)e_s = Z_\lambda e_s = z_{\lambda s}$. We may then rewrite eq. (3.4) in the form

$$[A_\lambda - (x + a_{\lambda s})e_s^{\mathrm{T}}]y = (A_\lambda + ce_s^{\mathrm{T}})y = g, \tag{3.6}$$

where

$$c = -x - a_{\lambda s}.$$

Using the orthogonal factorization, we have

$$Q[T_\lambda + Q^T c e_s^T Q]Q^T y = g,$$

giving

$$(T_\lambda + df^T)Q^T y = Q^T g, \tag{3.7}$$

where

$$d = Q^T c, \qquad f^T = e_s^T Q \qquad \text{and} \qquad g = r + y_s \tilde{y}.$$

The matrix df^T is a rank one modification of the quasi-triangular matrix T_λ. To solve this system, we need to retriangularize $T_\lambda + df^T$. Accordingly, we premultiply the system by two orthogonal matrices Q_1 and Q_2, giving

$$Q_2 Q_1 (T_\lambda + df^T)Q^T y = Q_2 Q_1 Q^T g, \tag{3.8}$$

where Q_1 and Q_2 are products of elementary plane rotations determined as follows.

The matrix Q_1 is such that

$$Q_1 d = (P_2 P_3 \cdots P_n)d = \gamma e_1 \qquad \text{where} \quad \gamma = \| d \|_2 \tag{3.9}$$

and P_i is a rotation in the $(i-1, i)$ plane designed to annihilate the ith component of $P_{i+1}P_{i+2} \cdots P_n d$. We have

$$Q_1(T_\lambda + df^T) = Q_1 T_\lambda + \gamma e_1 f^T, \tag{3.10}$$

where $Q_1 T_\lambda$ is upper Hessenberg, while $\gamma e_1 f^T$ is null except in the first row. Hence the right-hand side is also an upper Hessenberg matrix H. H may now be reduced to upper triangular form, $\bar{T}_\lambda$, by premultiplication with Q_2 defined by

$$Q_2 = P_n' \cdots P_3' P_2', \tag{3.11}$$

where the premultiplication by P_i' annihilates the element $(i, i-1)$ of the current matrix by a rotation in the $(i-1, i)$ plane. Hence, the triangular system remains to be solved

$$\bar{T}_\lambda Q^T y = Q_2 Q_1 Q^T g. \tag{3.12}$$

A system with the matrix B may thus be solved in $O(n^2)$ operations.

By its nature, eq. (3.2), which leads to eq. (3.12), is mildly nonlinear. Thus we repeat the process with $x + \tilde{y}$ and $\lambda + \mu$ as the new approximations. The convergence theorem for this iterative procedure can be found in [1]. Since an orthogonal triangularization of A is available, it becomes practical to update the matrix B at each stage of the iteration, using the current approximation to the eigenpair. Accordingly, we treat the $(p+1)$th step of the iteration as though it were the first step in the basic iteration starting with values $\lambda^{(p)}$ and $x^{(p)}$. Since we treat each iterate as if it were the first, the term $\mu\tilde{y}$ in eq. (3.2) is zero. Thus $g = \lambda x - Ax$. The algorithm then becomes

$$(A - \lambda^{(p)} I)\tilde{y}^{(p)} - y_s^{(p)} x^{(p)} = r^{(p)} = (\lambda^{(p)} I - A)x^{(p)} \tag{3.13}$$

where

$$x^{(p+1)} = x^{(p)} + \tilde{y}^{(p)}, \qquad \lambda^{(p+1)} = \lambda^{(p)} + y_s^{(p)}.$$

We may rewrite this as

$$B^{(p)} y^{(p)} = r^{(p)}, \tag{3.14}$$

where

$$B^{(p)} = A - \lambda^{(p)} I + c^{(p)} e_s^T, \qquad c^{(p)} = -x - a_{\lambda s}^{(p)}.$$

We now write

$$B^{(p)} = Q(T_\lambda^{(p)} + Q^T c^{(p)} e_s^T Q)Q^T$$

$$= Q(T_\lambda^{(p)} + d^{(p)}f^T)Q^T \tag{3.15}$$

and solve (3.14) by the triangular system

$$\bar{T}_\lambda^{(p)}Q^T y^{(p)} = Q_2^{(p)}Q_1^{(p)}Q^T r^{(p)}. \tag{3.16}$$

Convergence is detected by examining successive values for the correction to the eigenvalue. When the previous correction is smaller by a factor of 2 than the current correction, the iteration is stopped.

Note that Q and f will be independent of p if s is not changing from one iteration to the next. The rotations involved in $Q_1^{(p)}$ and $Q_2^{(p)}$, on the other hand, differ from one iteration to the next; but because the number of operations in each retriangularization is $O(n^2)$ and since some $\frac{1}{2}n^2$ multiplication and additions are necessarily involved in the solution of a triangular system, this is quite acceptable.

If we examine the number of operations for the reduction to quasi-triangular form, the *pre*-**SICE** phase, we see that there are roughly $10n^3 + 30n^2$ operations involved (here, an operation means a multiplication followed by an addition). For the **SICE** phase there are approximately $13n^2$ operations per iteration. Assuming 3 iterations to improve an eigenpair, the total count is approximately $39n^2$ operations.

ACKNOWLEDGMENT

I would like to thank the referees for their valuable suggestions, which have improved the quality of the software.

REFERENCES

1. DONGARRA, J.J., MOLER, C.B., AND WILKINSON, J.H. Improving the accuracy of computed eigenvalues and eigenvectors. *SIAM J. Numer. Anal.* to appear (ANL-81-43, Argonne National Laboratory, Argonne, Ill. (1981)).
2. DONGARRA, J.J., BUNCH, J.R., MOLER, C.B., AND STEWART, G.W. *LINPACK Users' Guide.* SIAM Pub., Philadelphia, Pa., 1979.
3. LAWSON, C.L., HANSON, R.J., KINCAID, D.R., AND KROGH, F.T. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Softw. 5*, 3 (Sept. 1979), 308–323.
4. SMITH, B.T., ET AL. Matrix eigensystem routines—EISPACK guide. In *Lecture Notes in Computer Science*, 2nd ed., vol. 6, Springer-Verlag, Berlin, 1976.
5. SYMM, H.J., AND WILKINSON, J.H. Realistic error bounds for a simple eigenvalue and its associated eigenvector. *Numer. Math 35* (1980), 113–126.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service.]

```
      SUBROUTINE SICEDR(LD, N, A, T, Q, W, X, CW, CX, K, Y1, Y2, R, Z1, SIC   1Ø
     * WR, WI, JOB, INFO, INIT)                                          SIC   2Ø
C                                                                        SIC   3Ø
      INTEGER I, J, IP1, IERR, LD, N, K, JOB, INFO, INIT                 SIC   4Ø
      REAL A(LD,1), T(LD,1), Q(LD,1), WR(1), WI(1)                       SIC   5Ø
      REAL W, X(1), CW, CX(1), Y1(1), Y2(1), R(1), Z1(1)                 SIC   6Ø
C                                                                        SIC   7Ø
C     THIS SUBROUTINE PROVIDES TWO FUNCTIONS DEPENDING ON                SIC   8Ø
C     THE VALUE OF INIT.                                                 SIC   9Ø
C                                                                        SIC  1ØØ
C          1) TO REDUCE A MATRIX TO QUASI-TRIANGULAR FORM,               SIC  11Ø
C             ACCUMULATING THE ORTHOGONAL TRANSFORMATIONS,               SIC  12Ø
C             AND DETERMINE THE EIGENVALUES OF THE MATRIX.               SIC  13Ø
C                                                                        SIC  14Ø
C          2) TO IMPROVE THE ACCURACY OF AN EIGENVALUE AND               SIC  15Ø
C             IMPROVE THE EIGENVECTOR OR COMPUTE AND IMPROVE THE         SIC  16Ø
C             EIGENVECTOR, GIVEN THE ORIGINAL MATRIX, AN                 SIC  17Ø
C             APPROXIMATE EIGENVALUE, THE QUASI-TRIANGULAR MATRIX,       SIC  18Ø
C             AND THE ORTHOGONAL MATRIX WHICH PRODUCED THE               SIC  19Ø
C             QUASI-TRIANGULAR FORM.                                     SIC  2ØØ
C                                                                        SIC  21Ø
C     THESE FUNCTIONS ARE SIGNALED BY THE PARAMETER INIT.                SIC  22Ø
```

ALGORITHM 590
DSUBSP and EXCHQZ: FORTRAN Subroutines for Computing Deflating Subspaces with Specified Spectrum

P. VAN DOOREN
Stanford University

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*eigenvalues*; G.m [**Mathematics of Computing**]: Miscellaneous—*FORTRAN*

General Terms: Algorithms

Additional Key Words and Phrases: Generalized eigenvalue, *QZ* algorithm

1. DESCRIPTION

A reliable and widely available method to compute the generalized eigenvalues of an $n \times n$ real regular (i.e., invertible) pencil $\lambda B - A$ is the so-called QZ-algorithm [1, 3]. This algorithm constructs orthogonal row and column transformations Q_1 and Z_1 such that the transformed pencil

$$\lambda B_1 - A_1 = Q_1(\lambda B - A)Z_1 \tag{1}$$

is in "quasi-triangular" form, that is, with B_1 in upper triangular form and A_1 in block triangular Hessenberg form with 1×1 and 2×2 diagonal blocks, as illustrated below:

$$B_1 = \begin{bmatrix} x & x & x & x & x & x \\ & x & x & x & x & x \\ & & x & x & x & x \\ & & & x & x & x \\ & & & & x & x \\ & & & & & x \end{bmatrix}, \qquad A_1 = \begin{bmatrix} x & x & x & x & x & x \\ & x & x & x & x & x \\ & x & x & x & x & x \\ & & & x & x & x \\ & & & & x & x \\ & & & & x & x \end{bmatrix}. \tag{2}$$

The 1×1 diagonal pencils of $\lambda B_1 - A_1$ contain the real generalized eigenvalues of the pencil $\lambda B - A$, and the 2×2 diagonal pencils of $\lambda B_1 - A_1$ contain the complex generalized eigenvalues, a conjugate pair to each 2×2 pencil.

The FORTRAN subroutines **DSUBSP** and **EXCHQZ** allow one to update the decomposition (1) by postmultiplying Z_1 by Z_2 and premultiplying Q_1 by Q_2 such that

$$\lambda B_2 - A_2 = Q_2 Q_1(\lambda B - A)Z_1 Z_2 \tag{3}$$

Received 11 November 1981; revised 2 June 1982; accepted 2 August 1982
This work was supported by the National Science Foundation under Grant ENG78-10003 and by the U.S. Air Force under Grant AFOSR-79-0094.
Author's present address: Philips Research Laboratory, Avenue Van Becelaere 2, Box 9, B-1170, Brussels, Belgium.

is still in quasi-triangular form, but in addition, a specific ordering of generalized eigenvalues is obtained in this form. This is important in several applications [6, 7]. Indeed, when the generalized eigenvalues $\lambda_1, \ldots, \lambda_n$ are ordered such that $\lambda_1, \ldots, \lambda_k$ are inside a region Γ and $\lambda_{k+1}, \ldots, \lambda_n$ are outside this region, then the space spanned by the first k orthonormal columns of $Z = Z_1 Z_2$ is the deflating subspace [4] of $\lambda B - A$ corresponding to the spectrum inside Γ. In several applications, deflating subspaces have to be computed for different regions Γ. Of course, Γ has to be symmetric with respect to the real axis, since complex pairs of eigenvalues have to be categorized both outside or inside Γ in order for this deflating subspace to be real. No further assumptions have to be imposed on Γ: it can be open or not, connected or not.

The user has to provide a function describing the region Γ by testing whether the generalized eigenvalues of a 1×1 or 2×2 (diagonal) pencil lies inside or outside the region Γ. This function must be of the type

INTEGER FUNCTION FTEST (LSIZE, ALPHA, BETA, S, P)

with parameters:

LSIZE An integer containing the size of the considered pencil (1 or 2).

ALPHA,BETA Two real variables. In case **LSIZE**=1, the generalized eigenvalue of the considered pencil is given by **ALPHA/BETA**, which may be infinite when **BETA**=0.

S, P Two real variables. In case **LSIZE**=2, they contain the sum and product of the two complex conjugate generalized eigenvalues of the considered pencil.

FTEST The function value, which is put equal to 1 when the generalized eigenvalue(s) of the considered pencil is (are) inside the specified region Γ, and equal to --1 otherwise.

Simple examples for such routines are given by the functions **FIN**, **FOUT**, **FOLHP**, and **FCRHP**, describing the regions inside and outside the unit circle, the open left half-plane, and the closed right half-plane, respectively. Their listings are included below as templates for the user.

This routine is then used as parameter for the subroutine **DSUBSP**, which reorders the 1×1 and 2×2 diagonal pencils of the quasi-triangular form $\lambda B_1 - A_1$ such that those with generalized eigenvalues inside Γ appear first. The calling sequence for **DSUBSP** is

CALL DSUBSP (NMAX, N, A, B, Z, FTEST, EPS, NDIM, FAIL, IND)

with (parameters preceded by an asterisk are altered by the subroutine):

NMAX An integer containing the first dimension of the arrays **A**, **B**, and **Z**.

N An integer containing the current order of **A**, **B**, and **Z**.

;axA,*B Doubly subscripted real arrays containing the pencil to be reordered. On return, $\lambda B - A$ contains the final quasi-triangular pencil, rearranged with respect to the region Γ specified by **FTEST**.

***Z** A doubly subscripted real array into which the reducing column transformation is postmultiplied.

FTEST The integer function provided by the user to describe the region Γ of interest.

EPS A real number used as the convergence criterion. Maximal accuracy is obtained when **EPS** is set equal to relpr $\times \max\{\|\mathbf{A}\|_2, \|\mathbf{B}\|_2\}$, where relpr is the relative precision of the computer used. Smaller values of **EPS** will increase the amount of work without significantly improving the accuracy.

***NDIM** An integer giving the dimension of the computed deflating subspace.

***FAIL** A logical variable that on normal return is **.FALSE.** If the iterative part of the algorithm does not converge, **FAIL** is set to **.TRUE.**.

*IND An integer working array of dimension at least **N**.

DSUBSP is to be used together with the EISPACK programs **QZHES**, **QZIT**, and **QZVAL** [1] to reduce a full pencil $\lambda B - A$ to quasi-triangular form with the eigenvalues inside the contour Γ appearing first on diagonal. (For the explanation of the parameters **EPS1**, **IERR**, **ALPHAR**, **ALPHAI**, and **BETA** in these routines, see [1].):

```
CALL QZHES (NMAX, N, A, B, .TRUE., Z)
CALL QZIT (NMAX, N, A, B, EPS1, .TRUE., Z, IERR)
CALL QZVAL (NMAX, N, A, B, ALPHAR, ALPHAI, BETA, .TRUE., Z)
CALL DSUBSP (NMAX, N, A, B, Z, FTEST, EPS, NDIM, FAIL, IND)
```

Besides the function **FTEST**, describing the region Γ of interest, **DSUBSP** also uses the subroutines **EXCHQZ**, **GIV**, and **SROT**. **EXCHQZ** is a FORTRAN subroutine to interchange two adjacent (1×1 and/or 2×2) pencils of a quasi-triangular form. Specifically, it is supposed that A has a block of order $l1$ starting at the lth diagonal element, and a block of order $l2$ starting at the $(l + l1)$th diagonal element (illustrated below for $n = 5$, $l = 2$, $l1 = 2$, $l2 = 1$):

$$B = \begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \\ & & & & x \end{bmatrix}, \quad A = \begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \\ & & & & x \end{bmatrix}. \tag{4}$$

EXCHQZ constructs orthogonal row and column transformations V and W such that $V(\lambda B - A)W$ has consecutive blocks of order $l2$ and $l1$ at the lth diagonal element (illustrated for the example above):

$$VBW = \begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \\ & & & & x \end{bmatrix}, \quad VAW = \begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \\ & & & & x \end{bmatrix}. \tag{5}$$

The generalized eigenvalues associated with each diagonal block are interchanged along with the blocks. The column transformation W is postmultiplied into the array **Z**; the row transformation V is not stored since it is not required in the computation of bases of deflating subspaces.

The calling sequence for **EXCHQZ** is

```
CALL EXCHQZ (NMAX, N, A, B, Z, L, LS1, LS2, EPS, FAIL)
```

with (parameters preceded by an asterisk are altered by the subroutine):

NMAX An integer containing the first dimension of **A, B**, and **Z**.

N An integer containing the current order of **A, B**, and **Z**.

***A,*B** Doubly subscripted real arrays containing the pencil to be reordered.

***Z** A doubly subscripted real array into which the updating column transformation is postmultiplied.

L An integer containing the leading diagonal position of the first block to be interchanged.

LS1 An integer containing the size of the first block.

LS2 An integer containing the size of the second block.

EPS A convergence criterion (cf. **EPS** in the calling sequence of **DSUBSP**).

***FAIL** A logical variable that on normal return is **.FALSE.**. If the iterative part of the algorithm does not converge, **FAIL** is set to **.TRUE.**.

EXCHQZ requires the subroutines **GIV** and **SROT**, which are elementary routines that construct and perform Givens rotations on column or rows. **SROT** is a BLA subroutine and **GIV** is a modification of **SROTG** of the BLAS package [2] in order to allow a more compact code for **EXCHQZ**, which contains several calls to these routines.

2. METHOD AND PROGRAMMING DETAILS

The subroutine **DSUBSP** makes a first pass through the diagonal blocks of the quasi-triangular form $\lambda B - A$ in order to determine the sizes of the diagonal blocks and the locations of their generalized eigenvalues with respect to Γ. During this pass the integer vector **IND(•)** is created with entries ± 1 or ± 2. Here **sign [IND(I)]** refers to the location of the generalized eigenvalues of block **I** with respect to Γ (+ for inside Γ, − for outside Γ) and **abs[IND(I)]** indicates the size of this block. The entries of this vector are then rearranged such that the plus signs appear first. This is done using a "bubble sort," that is, each time a plus sign follows a minus sign, it is moved in front of all its preceding minus signs via consecutive permutations. Each permutation, of course, involves an interchanging of two consecutive blocks using the subroutine **EXCHQZ**.

EXCHQZ works in a fashion similar to the routine **EXCHNG**, developed for the reordering of the standard eigenvalue problem [5] (this is also the reason why an apparent parallelism with that paper has been pursued here). To interchange two consecutive blocks where at least one of them has order 1, a shift is performed to the real eigenvalue of a 1×1 block (in [5], this was only done for the interchange of two 1×1 blocks). Givens rotations are then constructed in a straightforward manner to interchange the shifted zero eigenvalue to the other block. The construction of the Givens transformations needed for the exchange of the blocks is done in such a way as to ensure backward stability.

In the case of two 2×2 blocks, an arbitrary QZ-step is performed on both blocks in order to eliminate the uncoupling between them. Then a sequence of QZ-steps using a previously determined shift is performed on both blocks. A decoupling with the blocks in the desired order is usually obtained with a few steps (rarely more than two). If within 30 iterations no decoupling is obtained, the subroutine gives an error return. The criterion used is that the coupling element of the two blocks in the Hessenberg form of A is smaller than **EPS**. Since it does not make sense to force this coupling element to be smaller than the errors in the rest of the pencil, one should not choose **EPS** smaller than relpr $\times$ $\max\{\|A\|_2, \|B\|_2\}$, where relpr is the relative precision of the computer used. A good choice for **EPS** is the "estimated" absolute precision of the (sometimes measured) data in A and B. Since the pencils used here are of dimension 4×4, the QZ steps are implemented with Givens transformations instead of Householder transformations, which turns out to be economical. More details are given in [7], where a proof of the backward stability of the method is also given.

EXCHQZ uses the routines **GIV** and **SROT**, which construct a 2×2 Givens rotation to zero out an element of a 2-vector, and perform it, respectively, on two columns or rows of a specified matrix.

ACKNOWLEDGMENTS

The valuable comments of Sven Hammarling and the helpful guidance distilled from Algorithms 506 and 539 are gratefully acknowledged.

REFERENCES

1. GARBOW, B.S., BOYLE, J.M., DONGARRA, J.J., AND MOLER, C.B. Matrix Eigensystem Routines—EISPACK Guide Extension. In *Lecture Notes in Computer Science*, vol. 51, Springer, New York, 1977.

2. LAWSON, C.L., HANSON, R.J., KINCAID, D.R., AND KROGH, F.T. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Softw. 5*, 3 (Sept. 1979), 324-325.
3. MOLER, C.B., AND STEWART, G.W. An algorithm for the generalized eigenvalue problem. *SIAM J. Numer. Anal. 10* (1973), 241-256.
4. STEWART, G.W. Error and perturbation bounds for subspaces associated with certain eigenvalue problems. *SIAM Rev. 15* (1973), 727-764.
5. STEWART, G.W. HQR3 and EXCHNG. Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix. *ACM Trans. Math. Softw. 2*, 3 (Sept. 1976), 275-280.
6. VAN DOOREN, P. The generalized eigenstructure problem in linear system theory. *IEEE Trans. Automat. Contr. AC-26* (1981), 111-129.
7. Van Dooren, P. A generalized eigenvalue approach for solving Riccati equations. *SIAM J. Stat. Sci. Comput. 2* (1981), 121-135.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service.]

```
      SUBROUTINE DSUBSP(NMAX, N, A, B, Z, FTEST, EPS, NDIM, FAIL, IND)  DSU   1Ø
      INTEGER NMAX, N, FTEST, NDIM, IND(N)                              DSU   2Ø
      LOGICAL FAIL                                                      DSU   3Ø
      REAL A(NMAX,N), B(NMAX,N), Z(NMAX,N), EPS                         DSU   4Ø
C*                                                                      DSU   5Ø
C* GIVEN THE UPPER TRIANGULAR MATRIX B AND UPPER HESSENBERG MATRIX A    DSU   6Ø
C* WITH 1X1 OR 2X2 DIAGONAL BLOCKS, THIS ROUTINE REORDERS THE DIAGONAL  DSU   7Ø
C* BLOCKS ALONG WITH THEIR GENERALIZED EIGENVALUES BY CONSTRUCTING EQUI-DSU   8Ø
C* VALENCE TRANSFORMATIONS QT AND ZT. THE ROW TRANSFORMATION ZT IS ALSO DSU   9Ø
C* PERFORMED ON THE GIVEN (INITIAL) TRANSFORMATION Z (RESULTING FROM A  DSU  1ØØ
C* POSSIBLE PREVIOUS STEP OR INITIALIZED WITH THE IDENTITY MATRIX).     DSU  11Ø
C* AFTER REORDERING, THE EIGENVALUES INSIDE THE REGION SPECIFIED BY THE DSU  12Ø
C* FUNCTION FTEST APPEAR AT THE TOP. IF NDIM IS THEIR NUMBER THEN THE   DSU  13Ø
C* NDIM FIRST COLUMNS OF Z SPAN THE REQUESTED SUBSPACE. DSUBSP REQUIRES DSU  14Ø
C* THE SUBROUTINE EXCHQZ AND THE INTEGER FUNCTION FTEST WHICH HAS TO BE DSU  15Ø
C* PROVIDED BY THE USER. THE PARAMETERS IN THE CALLING SEQUENCE ARE :   DSU  16Ø
C* (STARRED PARAMETERS ARE ALTERED BY THE SUBROUTINE)                    DSU  17Ø
C*                                                                      DSU  18Ø
C*    NMAX      THE FIRST DIMENSION OF A, B AND Z                       DSU  19Ø
C*    N         THE ORDER OF A, B AND Z                                 DSU  2ØØ
C*   *A,*B      THE MATRIX PAIR WHOSE BLOCKS ARE TO BE REORDERED.       DSU  21Ø
C*   *Z         UPON RETURN THIS ARRAY IS MULTIPLIED BY THE COLUMN      DSU  22Ø
C*              TRANSFORMATION ZT.                                      DSU  23Ø
C*    FTEST(LS,ALPHA,BETA,S,P) AN INTEGER FUNCTION DESCRIBING THE       DSU  24Ø
C*              SPECTRUM OF THE DEFLATING SUBSPACE TO BE COMPUTED:      DSU  25Ø
C*              WHEN LS=1 FTEST CHECKS IF ALPHA/BETA IS IN THAT SPECTRUM DSU  26Ø
C*              WHEN LS=2 FTEST CHECKS IF THE TWO COMPLEX CONJUGATE     DSU  27Ø
C*              ROOTS WITH SUM S AND PRODUCT P ARE IN THAT SPECTRUM     DSU  28Ø
C*              IF THE ANSWER IS POSITIVE, FTEST=1, OTHERWISE FTEST=-1  DSU  29Ø
C*    EPS       THE REQUIRED ABSOLUTE ACCURACY OF THE RESULT            DSU  3ØØ
C*   *NDIM      AN INTEGER GIVING THE DIMENSION OF THE COMPUTED         DSU  31Ø
C*              DEFLATING SUBSPACE                                      DSU  32Ø
C*   *FAIL      A LOGICAL VARIABLE WHICH IS FALSE ON A NORMAL RETURN,   DSU  33Ø
C*              TRUE OTHERWISE (WHEN EXCHQZ FAILS)                      DSU  34Ø
C*   *IND       AN INTEGER WORKING ARRAY OF DIMENSION AT LEAST N        DSU  35Ø
C*                                                                      DSU  36Ø
      INTEGER L, LS, LS1, LS2, L1, LL, NUM, IS, L2I, L2K, I, K, II,     DSU  37Ø
     * ISTEP, IFIRST                                                    DSU  38Ø
      REAL S, P, D, ALPHA, BETA                                         DSU  39Ø
      FAIL = .TRUE.                                                     DSU  4ØØ
      NDIM = Ø                                                          DSU  41Ø
      NUM = Ø                                                           DSU  42Ø
      L = Ø                                                             DSU  43Ø
      LS = 1                                                            DSU  44Ø
C*** CONSTRUCT ARRAY IND(I) WHERE :                                     DSU  45Ø
C***     IABS(IND(I)) IS THE SIZE OF THE BLOCK I                        DSU  46Ø
C***     SIGN(IND(I)) INDICATES THE LOCATION OF ITS EIGENVALUES         DSU  47Ø
C***                  (AS DETERMINED BY FTEST).                         DSU  48Ø
C*** NUM IS THE NUMBER OF ELEMENTS IN THIS ARRAY                        DSU  49Ø
      DO 3Ø LL=1,N                                                      DSU  5ØØ
        L = L + LS                                                      DSU  51Ø
```

```
      IF (L.GT.N) GO TO 4Ø                                                DSU  52Ø
      L1 = L + 1                                                          DSU  53Ø
      IF (L1.GT.N) GO TO 1Ø                                               DSU  54Ø
      IF (A(L1,L).EQ.Ø.) GO TO 1Ø                                         DSU  55Ø
C* HERE A 2X2  BLOCK IS CHECKED *                                         DSU  56Ø
      LS = 2                                                              DSU  57Ø
      D = B(L,L)*B(L1,L1)                                                 DSU  58Ø
      S = (A(L,L)*B(L1,L1)+A(L1,L1)*B(L,L)-A(L1,L)*B(L,L1))/D             DSU  59Ø
      P = (A(L,L)*A(L1,L1)-A(L,L1)*A(L1,L))/D                             DSU  6ØØ
      IS = FTEST(LS,ALPHA,BETA,S,P)                                       DSU  61Ø
      GO TO 2Ø                                                            DSU  62Ø
C* HERE A 1X1  BLOCK IS CHECKED *                                         DSU  63Ø
   1Ø LS = 1                                                              DSU  64Ø
      IS = FTEST(LS,A(L,L),B(L,L),S,P)                                    DSU  65Ø
   2Ø NUM = NUM + 1                                                       DSU  66Ø
      IF (IS.EQ.1) NDIM = NDIM + LS                                       DSU  67Ø
      IND(NUM) = LS*IS                                                    DSU  68Ø
   3Ø CONTINUE                                                            DSU  69Ø
C***  REORDER BLOCKS SUCH THAT THOSE WITH POSITIVE VALUE                  DSU  7ØØ
C***    OF IND(.) APPEAR FIRST.                                           DSU  71Ø
```

ALGORITHM 591
A Comprehensive, Matrix-Free Algorithm for Analysis of Variance

WILLIAM J. HEMMERLE
University of Rhode Island

Categories and Subject Descriptors: G.3 [**Mathematics of Computing**]: Probability and Statistics—*statistical computing*; *statistical software*; G.m **[Mathematics of Computing]**: Miscellaneous—*FORTRAN*

General Terms: Algorithms

Additional Key Words and Phrases: Linear models, analysis of variance, unbalanced data, missing cells, estimation, hypothesis testing, storage-efficient algorithm, iterative improvement, balanced data operators, algebraic-model specification, rank computations, G-inverse solution.

1. INTRODUCTION

Some of the work to be described is reminiscent of **AARDVARK** [5], an analysis-of-variance package, at Iowa State University in the early 1960s. This program was used for most of the large number of analyses processed at Iowa State at that time; however, the program was definitely not transportable, being written in a mix of assembler and FORTRAN for a nonstandard IBM 7074. Subsequent versions, lacking some of its initial capabilities but adding others, were prepared for other machines, including the IBM 360 (see for example [7]). One distinctive feature of the program that greatly aided in its usability was algebraic specification of the statistical model by the user [14], [18]. This facility has now become common in statistical packages.

The initial **AARDVARK** relied principally on balanced analysis-of-variance (AOV) algorithms and approximate statistical methods were applied to treat unbalanced data. An iterative AOV algorithm was later developed by Hemmerle [12], which permits using balanced analysis-of-variance algorithms to obtain exact statistical results for unbalanced data. This algorithm along with subsequent work will be exploited in our development of a comprehensive analysis-of-variance algorithm for balanced or unbalanced data.

Balanced analysis-of-variance algorithms frequently represent a level of elegance of logic that is difficult to surpass. One fine example is the algorithm due to H. O. Hartley [3], which obtains a complete factorial decomposition. Another is Wilkinson's recursive algorithm [23]. Unfortunately, these algorithms are also frequently difficult to explain using straightforward notation; they are likely to involve matrices with complicated structures implicitly rather than explicitly.

Received 27 February 1979; revised 17 May 1982; accepted 26 May 1982
This research was partially supported by the National Science Foundation under Grants DCR74-15331 and MCS 74-15331 A01 and by the Department of Statistics, North Carolina State University during a period of sabbatical leave.
Author's address: Department of Computer Science and Experimental Statistics, University of Rhode Island, Kingston, RI 02881.

Balanced analysis-of-variance algorithms normally do not require the coding of indicator variables, either externally or internally by the program. They possess the virtue of efficiency, both with respect to the number of operations performed and the amount of storage required.

Data, on the other hand, are frequently unbalanced and are likely to have missing cells. Both theoretically and computationally, the order of complexity of analysis-of-variance problems increases down the following categories:

(1) balanced data;
(2) unbalanced data with no missing cells;
(3) unbalanced data with missing cells.

In some cases, the unbalanced data arise from a balanced design in which some of the observations are missing as a result of the experiment. Computationally, there are a number of missing data algorithms, notably Healy and Westmacott's iterative algorithm [4] and Rubin's noniterative algorithm [16], which essentially fill in the missing values for use with balanced algorithms. One could also use dummy covariates for each missing cell and then compute a balanced analysis of covariance. These approaches have the disadvantage that, in general, they require balancing all of the data (including the missing observations) to have equal cell frequencies.

Almost all algorithms which seek to process unbalanced data with any degree of generality in an analysis-of-variance situation, including those in statistical packages, are matrix-oriented least squares or regression algorithms. These involve the creation of indicator variables, to cast the analysis-of-variance model as a regression model, and the formation of the design matrix X_0 or the coefficients matrix of the normal equations $X_0'X_0$. The normal equations are then solved using either orthogonalization or elimination methods. In some cases, SAS Procedure GLM [1], for example, an explicit G-inverse is obtained for $X_0'X_0$. One problem with the matrix-oriented approach is that the dimensions of X_0 or $X_0'X_0$ can be overwhelming for moderate to large analysis-of-variance problems. This approach is also extremely inefficient when the data are balanced.

2. FEATURES AND DESIGN OBJECTIVES

In what follows, we describe a global algorithm for the analysis of variance with the following features.

(1) Balanced data, unbalanced data, and unbalanced data with missing cells are all processed by the algorithm. This is accomplished without losing the operational efficiencies obtainable from balanced data and without applying approximate statistical methods to the unbalanced data.

(2) The algorithm is very general with respect to the kind of problems it can handle. Specifically, it bases its calculations upon an algebraically specified analysis-of-variance model of the type discussed in Searle [19]. This includes models with crossed factors, nested factors, and interactions between factors. This general model, along with the facility to handle missing cells, also includes such designs as incomplete blocks, lattices, and Latin squares.

(3) Very large problems may be processed using a relatively small amount of computer storage. With one minor exception, no matrices are stored and no explicit matrix operations are performed. In particular, neither X_0 nor $X_0'X_0$ is stored or computed. An exact G-inverse solution to the normal equations is obtained without ever computing a G-inverse. The rank of the design matrix X_0 is obtained from the pattern of missing cells without explicit operations on X_0.

(4) The algorithm provides a heuristic optimum of maximizing analysis-of-variance capabilities while minimizing lines of source code. Since a principal attribute of the algorithm is its limited demands for array storage, an effort was made to also constrain program storage through a judicious selection of functional capabilities and the production of tight code. Consequently, the algorithm should be well suited for use on small computers.

(5) The user is given reasonable facilities in specifying his analysis-of-variance problem algebraically and various options alphanumerically in an essentially format-free manner.

(6) The algorithm is well suited to interactive usage. It was developed using **CALL/OS** at the University of Rhode Island and **TSO** (time-sharing option) at North Carolina State. It may, of course, also be used as a batch program. A mode parameter, interactive or batch, determines the nature of prompting, I/O, and error returns.

(7) The program has been written in American National Standard Basic FORTRAN for portability. The program has been verified by the Bell Laboratories PFORT verifier [17] and should run on any computer supported with a FORTRAN 66 compiler. With the exception of using the Hollerith data type in DATA statements, the program is also compatible with FORTRAN 77.

Along with the effort to constrain program storage, steps were taken to reduce the number of *lines* of source code. Lines of source (including **COMMENT** statements) can be a limiting factor with respect to file storage in interactive systems. Most of the following measures served to reduce program storage as well as lines of source code: limiting the number of specification statements, **WRITE** statements, **FORMAT** statements, and unnecessary **CONTINUE** statements; limiting the amount of constant alphanumeric information in **FORMAT** statements; using selective, concise **COMMENT** statements; accepting trivial operational inefficiencies which reduce lines of source code; avoiding modularity not contributing to operational efficiency or storage reduction; and using single-character identifiers when possible. The portability objective essentially limits one to storage of a single alphanumeric character per alphanumeric constant or array location. Rather than use additional code to interpret multiple-character identifiers, for such things as specified options, we limit these to one character.

As a consequence of these measures, the complete program for the global algorithm consists of 836 FORTRAN statements exclusive of **COMMENT** statements. The object code produced by the **WATFIV** compiler was contained in 36,656 bytes. In addition, the program was written in such a manner that several of the functional capabilities may easily be deleted, if necessary, to relax program storage.

3. CONSTITUENT ALGORITHMS

We have called the algorithm under discussion a global algorithm, since it encompasses the use of several independent algorithms. The principal independent algorithms, which comprise the global algorithm, are the following.

3.1 The Iterative AOV Algorithm

This algorithm iteratively applies expectation and residual operators (E/R operators) for balanced data to obtain exact results for unbalanced data. In other words, stated simply, the algorithm computes an analysis of variance for unbalanced data by successive computation of balanced analyses. Since the algorithm uses balanced E/R operators to obtain its results, there is no need to create indicator variables or form the design matrix X_0 or $X_0'X_0$. The algorithm operates upon a vector of cell sums and a vector of cell frequencies rather than upon a (potentially large) matrix so that array storage requirements are minimal. Missing cells are handled routinely and convergence of the algorithm is guaranteed [12]. A useful monotonic property serves to limit iteration when testing hypotheses. A balanced analysis of variance is a special case requiring only one iteration.

The basic algorithm [12] is very general and applies to any analysis of variance; however, one must know or determine the balanced E/R operators. It is in this context that one can apply much of the algorithmic lore associated with balanced analyses. An initial implementation using a relatively comprehensive class of balanced E/R operators was discussed by Hemmerle and Piette [6]. Many of the features of the global algorithm were included in this implementation; however,

certain unnecessary rank restrictions were placed upon the design matrix X_0. The latter restrictions were subsequently removed by Hemmerle [10], who showed that with proper parameter selection the iterative AOV algorithm would obtain a G-inverse solution to the normal equations for any design matrix. An improved iterative approximation, which halves the number of iterations in testing hypotheses, is also given in [10] and is used here.

An iterative analysis of covariance algorithm was also developed by Hemmerle [11] as an extension of the iterative AOV algorithm. Although this covariance algorithm adapts well to the logic of the global algorithm and is array storage efficient, covariance capabilities were excluded here principally because of the additional program storage requirements.

3.2 An Iterative Rank Algorithm

With unbalanced data, including missing cells, the ranks of the design matrix and restricted design matrix must be computed to determine degrees of freedom for F statistics. Algorithms have been included (Section 3.5) to determine rank noniteratively from the pattern of missing cells for a given model when possible. When this cannot be accomplished, a recent adaptation of the iterative AOV algorithm is used to compute rank in one of two ways, depending upon the number of missing cells. Either way will produce monotonically increasing convergence to the integer rank value. For a small number of missing cells we use some matrix storage (the exception cited earlier) and obtain monotonically increasing quadratic convergence. Both methods are outlined in a subsequent section. A complete discussion is given in [9].

3.3 A Balanced Factorial Decomposition Algorithm

The technique for computing a balanced analysis of variance based upon a factorial decomposition of linear combinations of classifications means dates back to H. O. Hartley [3].

We use the algorithm described in Hemmerle [13, pp. 180–185] to obtain this decomposition. A one-to-one correspondence is maintained between the distinct arrays in this decomposition and the locations of an E/R list which we shall discuss shortly.

3.4 A Pooling Algorithm

In order to construct the E/R operators needed for the iterative AOV algorithms, we must "pool" (add together in the proper manner) distinct arrays in the factorial decomposition as dictated by the particular analysis-of-variance model. The complete factorial decomposition will be stored in a single linear, one-dimensional array, and we must be able to map one of the distinct arrays within this decomposition into another. We use essentially the same algorithm described in Schlater and Hemmerle [18] to do the mapping and pooling; however, we pool arrays of the decomposition rather than classification means, as do the latter.

3.5 Algorithms to Recognize Balance and Restructure Data

A balanced analysis of variance is a special case for the iterative AOV algorithm. When properly executed, final results will be obtained on the first iteration. Furthermore, iterative rank computations are unnecessary. With unbalanced data, there are also frequent situations in which some of the results may be obtained noniteratively. An example would be a full factorial model; the rank of the design matrix, the sum of squares for error (SSE), the sum of squares for regression (SSR), and a G-inverse solution to the normal equations may all be obtained from a balanced analysis of variance on cell means [12]. Also, when testing an hypothesis involving unbalanced data, the data structure for the model restricted by the hypothesis frequently has fewer dimensions than it has for the full model. A simple restructuring of the data may facilitate obtaining results without iterating. Essentially, this involves creating a surrogate vector of cell

frequencies based upon reduced dimensions for the same data; however, duplicate entries are made in this surrogate vector to maintain a proper one-to-one correspondence with the original vector of cell sums. Several algorithms have been incorporated into the global algorithm to recognize balance and restructure data, when appropriate, to avoid unnecessary iteration. The related theory and a description of the algorithms will be found in [8].

3.6 An E/R List Construction Algorithm

Numeric calculations in the global algorithm are driven by what we have called an E/R list. If there are n factors in the full analysis-of-variance model, then the E/R list will have 2^n entries corresponding to the 2^n terms (mean, main effects, and interactions) in a full factorial model. The E/R list is constructed by scanning or parsing an algebraic model statement. Numeric entries are made in the list which uniquely describe the model. Following the procedure suggested by Hemmerle [14], decreasing powers of 2 are assigned as numerical values to factor symbols and their associated subscripts, with unity for the last factor and its associated subscript.

The sum of the numerical values of the factor symbols (plus one) is computed for each term in the model. For crossed factors or interactions between crossed factors, this sum is entered in the corresponding location of the E/R list. For nested factors or interactions involving nested factors, multiple entries are made in the E/R list. The sum is entered into each location of the E/R list which corresponds to an array in the factorial decomposition which must be pooled (see [13, pp. 174–176]). The algorithm determines these locations from the numerical values of the factor symbols and subscripts in the model term. Examples will be given later in the sequel.

An hypothesis statement indicates those terms in the full model which should be deleted in forming the reduced model. As this statement is parsed, the E/R list entries for the model terms in the statement are made negative. The R operator is then always formed by pooling the arrays in the factorial decomposition associated with zero or negative E/R list entries.

4. STATISTICAL FOUNDATIONS AND COMPUTATIONS

Analysis-of-variance models tend to create confusion because of the fact that in the form they are usually written, these models are overparameterized. For example, the two-way classification with interaction is usually written as

$$y_{ijk} = \mu + a_i + b_j + ab_{ij} + e_{ijk},$$
$$i = 1, \ldots, I, \quad j = 1, \ldots, J, \quad k = 1, \ldots, n_{ij}. \quad (4.1)$$

With e_{ijk} distributed $N(0, \sigma^2)$ and σ^2 unknown. None of the parameters μ, a_i, b_j, or ab_{ij} in (4.1) are estimable (for any i or j), even though the data may be balanced. That is, although the normal equations for (4.1) are consistent, there is no unique solution for these equations. We may obtain a unique solution by imposing constraints or side conditions upon the solution. These conditions have no effect upon what is estimable from the data for the model; however, they do have a bearing upon the hypothesis that is tested by applying the same applicable conditions to a reduced model.

Speed and Hocking [22] discuss the resulting difference in the application of Searle's R ( ) notation [19] to the initial, overparameterized model as opposed to a reparameterized model obtained by imposing conditions on the original model. They refer to the latter as R* ( ) or procedure 2. Speed et al. [21] survey the hypotheses being tested by current computational methods and computer programs. A significant by-product of this work is that, when using procedure 2, the unweighted summation constraints (e.g., $\sum_i a_i = \sum_j b_j = \sum_i ab_{ij} = \sum_j ab_{ij} = 0$) yield the most likely hypotheses of interest, the classical Yates hypotheses [24].

The global algorithm uses procedure 2 with the unweighted summation conditions. In matrix form, the full reparameterized model is given by

$$y = X_0\beta + e \tag{4.2}$$

where y is a vector of N observations; X_0 is an $N \times p$ matrix of -1, 0, 1 indicator variables consistent with the unweighted summation conditions; β is a vector of p parameters; and the elements of e are distributed $N(0, \sigma^2)$.

Note that the p parameters in β will be linear combinations of the parameters appearing in the overparameterized model. As an example, consider the two-way classification *without* interaction. We have that

$$\beta' = (\mu + a. + b., a_1 - a., \ldots, a_{I-1} - a., b_1 - b., \ldots, b_{J-1} - b.) \tag{4.3}$$

with the dot notation denoting a mean. If X_0 has full rank, then all of the elements of β are linearly independent estimable functions. A fact that sometimes causes confusion is that $a_I - a.$ and $b_J - b.$ are also estimable in the above example; however, they are not linearly independent of the elements in β. We mention this fact here since the global algorithm yields a G-inverse solution for the initial overparameterized model as well as a G-inverse solution for (4.2). These solutions are the same, except for the inclusion of additional nonlinear independent elements in the former solution.

In order to consider hypothesis testing, we partition the design matrix X_0 for the full model as $[X_{01} \mid X_{02}]$, where X_{01} is $N \times k$, and write the model (4.2) as

$$y = X_{01}\beta_1 + X_{02}\beta_2 + e. \tag{4.4}$$

The global algorithm will attempt to test the hypothesis

$$H_0 : \beta_2 = 0 \tag{4.5}$$

by fitting the reduced model

$$y = X_{01}\beta_1 + e. \tag{4.6}$$

The hypothesis statement specifies the model terms which are to be deleted from the full model in forming the reduced model (4.6). For balanced data or unbalanced data with no missing cells, these are the same hypotheses tested, with F statistics, in a standard balanced analysis-of-variance table. A rule is given in [9] to determine whether or not the equivalent balanced data hypothesis is testable for unbalanced data with missing cells. For a model with all factors crossed, this rule is simply: the equivalent balanced data hypothesis is testable if and only if the rank of the full model minus the rank of the reduced model equals the degrees of freedom one would use for the hypothesis with balanced data. It is the author's opinion that when the equivalent balanced data hypothesis is not testable, those hypotheses that are testable are usually very difficult to interpret in a meaningful way.

Some designs which are considered to be balanced designs will require iteration because the residual operator used applies to a more general model. An example of this is a Latin square which the algorithm treats as a missing cell problem. Iteration will be necessary to obtain SSE for the full model; however, as a result of data restructuring, iteration is not required in testing the relevant hypotheses.

The following statistics may be obtained from the algorithm, most of them on an optional basis:

(1) cell sums, frequencies, and means;
(2) classification sums, frequencies, and means;
(3) rank (X_0);
(4) SSE (full model) $= y'[I - X_0(X_0'X_0)^-X_0']y$ (invariant);
(5) SSR (full model) $= y'X_0(X_0'X_0)^-X_0'y$;
(6) a G-inverse solution to $X_0'X_0\hat{\beta} = X_0'y$; $\hat{\beta} = (X_0'X_0)^-X_0'y$;
(7) estimates of expected cell means $= X_0\hat{\beta}$ (invariant);
(8) rank (X_{01});

(9) SSE (reduced model) $= y'[I - X_{01}(X'_{01}X_{01})^{-}X'_{01}]y$;

(10) SSR (reduced model) $= y'X_{01}(X'_{01}X_{01})^{-}X'_{01}y$;

(11) $$F = \frac{(\text{SSR(full)} - \text{SSR(reduced)})/(\text{rank}(X_0) - \text{rank}(X_{01}))}{\text{SSE(full)}/(N - \text{rank}(X_0))};$$

(12) probability values; Prob$\{F \mid H_0\}$.

We indicated earlier that a G-inverse solution is obtained for the overparameterized model as well. It should be clear that the rank of X_0 is the same as the rank of the design matrix of the overparameterized model formed using 0, 1 indicator variables.

5. LOGICAL COMPONENTS

The program for the global algorithm consists of a main program and eight subroutines. The principal function of these components is described below.

MAINA processes the factor, level statement; computes cell sums, cell frequencies, and $y'y$ as it reads the data; processes the options statement and contains much of the code to execute the A (cell means) and C (classification means) options.

SCAN processes the model and hypothesis statement to construct (or modify) the E/R list; computes parameters needed in restructuring data; computes the degrees of freedom applicable to data with no missing cells.

IGET is used by the main program and subroutine **SCAN** to sequentially retrieve characters (other than blank, plus, or comma) from the input buffer.

PART1 restructures the data (cell frequencies) when appropriate; checks for balance and alternative noniterative computations; computes rank noniteratively if possible or iteratively otherwise when the R (rank) option is specified.

PART2 is the principal numeric computational component; with the use of the remaining four subroutines, it computes SSE, SSR, estimates of expected cell means, a G-inverse solution to the normal equations, F statistics, and probability values.

STEP performs one basic step of the iterative AOV algorithm [12] using the improved approximation for SSR given in [10]. This basic step consists of the following substeps.

(1) $A \leftarrow (Y - D \cdot V)/c$
(2) $V \leftarrow V + A$
(3) $B \leftarrow B + A$
(4) $A \leftarrow R[A]$
(5) $V \leftarrow V - A$
(6) $S \leftarrow 2 \cdot Y'V - V'DV$

where

Y is the vector of cell sums, D is the diagonal matrix of cell frequencies stored as a vector, and c is an algorithm constant set to assure convergence and, in hypothesis testing, to assure monotonicity of the approximation S to SSR;

A, B, and V are work vectors of size n_c, with n_c being the number of cells. An additional work vector large enough to contain the factorial decomposition cited earlier is also required; however, vector A will occupy the first n_c locations of this additional vector;

$R[A]$ is the residual operator applied to vector A. Substep (4) above consists of a factorial decomposition of vector A, followed by pooling the appropriate arrays of the decomposition back into A.

The vectors B and V must be initially set to zero. Upon completion, perhaps after many steps, estimates of the expected cell means will be contained in V, and a G-inverse solution is obtained by applying the E operator to the vector B.

Substep (1) above is modified slightly for iterative rank computations and for restructured data. In computing rank, this takes the form

$$A \leftarrow (Y - \Delta V),$$

where Δ is a diagonal matrix with unit diagonals for filled cells and zero diagonals for missing cells. Operations with Δ are handled implicitly and this matrix is never stored. In performing noniterative calculations (a single step) on restructured cell frequencies, substep (1) is modified to

$$A \leftarrow (D2^{-} \cdot Y - V)$$

where $D2^{-}$ is a diagonal G-inverse of the diagonal matrix of restructured cell frequencies [8].

Substep (6) above must also be modified in computing rank iteratively with Δ replacing D. Furthermore, in rank computations, the vector Y contains a dummy unit vector; cell sums have been temporarily stored elsewhere in a vacant vector.

DECOMP obtains a factorial decomposition of a given vector; determines the classification frequencies needed in **PART1** to restructure data; computes classification means for the C option in **MAINA**.

POOL either moves the secondary array into the primary array, duplicating entries where needed, or it pools the secondary array into the primary array by addition.

LABEL calculates the array of coefficients for the array map needed in pooling; produces output labels for classification means and for the G-inverse solution.

Of these logical components, all of **PART1** may be deleted to further reduce program storage without making **PART2** inoperable; however, this would make the program less efficient owing to unnecessary iteration and full rank would be assumed by default for X_0 or X_{01}. Almost all of **SCAN** can also be deleted, provided the user constructs and inputs the required E/R list.

6. ARRAY STORAGE

We previously emphasized the storage efficiency of the algorithm for moderate to large problems. Fundamentally, this efficiency is attributable to the use of the iterative AOV algorithm; however, array storage economy was also stressed in constructing the global algorithm. The following is a brief description of all of the arrays included.

LSTFI is the array discussed in [13] and [18], which is used in formation and subsequent manipulation of the arrays in the factorial decomposition; this is an array of size 2^n, where n is the number of factors.

LER is the E/R list that is also of size 2^n.

LE is an alphanumeric array for factor symbols of size n.

LS is an alphanumeric array of associated subscripts of size n.

LV is an array of the numerical values assigned to the factor symbols and their associated subscripts; it also has size n.

LLIM contains the number of levels for each factor and is of size n.

LT is a temporary work array of size n.

LP is also a work array; however, its size is 10, the maximum number of factors handled by the program.

LD is an alphanumeric array of size 10 containing the digits 0–9; this array is used to convert alphanumeric input, such as the number of levels for a factor, to numeric.

LO is an alphanumeric array of size 10 containing the single-character option identifiers.

IA is an alphanumeric input buffer—a card image--which is used in processing the factor, levels, options, model, and hypothesis statements and also for storage

Table I. Options

Identifier	Function of option	Default
S(d)	Significant digits in results	$d = 5$
T($l \times 10^4$)	Test level	$l = 0.05$
I(m)	Iteration maximum	$m = 100$
R	Rank computations (iterative)	off
V	Estimates of expected cell means	off
G	G-inverse solution to normal equations	off
P	Probability values for F statistics	off
Z	Intermediate output	off
A	Cell sums, frequencies, and means	off[a]
C	Classification sums, frequencies, and means	off[a]

[a] These switches do not change their status each time the option is specified.

of a variable **FORMAT** statement for input data. (The model and hypothesis statements have continuation facilities.)

Q is the only two-dimensional array. It is used exclusively for rank computations and its size, in relation to the number of missing cells, determines the rank algorithm applied. If the dimension of **Q** were fixed at 10 × 10, then the quadratically convergent rank algorithm would be used only if there were no more than 10 missing cells.

QT is a vector whose size is the same as one of the dimensions of **Q**; it is used in conjunction with **Q** in computing rank.

W is a linear array that is used for all the numeric computations. This array logically consists of six contiguous vectors. The implicit names of these vectors coincide with those used in describing a step of the iterative AOV algorithm; they are ordered in storage as

Y, D, D2, V, B, A.

The first five of these vectors have size n_c, where n_c is the number of cells. The sixth vector is the size of the factorial decomposition for the problem. For example, with two factors we have n_c = **I•J**, while the vector **A** has size **(I + 1)(J + 1)**. Ordinarily, the array **W** would be allocated all of the remaining computer storage; the implicit variable dimensioning of **W** maximizes the number of analysis-of-variance problems that can be processed within a given amount of storage.

7. THE FACTOR, LEVELS STATEMENT

The factor, levels statement creates entries for the arrays **LE, LS, LV,** and **LLIM.** The array **LSTFI** is then formed from **LLIM.** Factor symbols, subscript symbols, and factor levels are included in the statement; the number of factors and their numerical values are determined. As an example of a factor, levels statement, consider an analysis involving 3 factors with levels 5, 10, and 15. If we decide to name the factors **R, C, T** with subscripts **I, J, K,** then the statement would be written (typed/punched) on one line as

F(R, C, T), L(I(5), J(10), K(15))

All of the commas above are cosmetic, as are blanks and pluses; the same statement could be written as

F(RCT)L(I(5)J(10)K(15))

8. OPTIONS AND THE OPTIONS STATEMENT

A list of option identifiers and default values is given in Table I. The first three identifiers include an argument enclosed within parentheses which are used to modify a parameter. The remaining seven identifiers trigger on/off switches; with the exception of the A and C option, these switches change their status each time the option is specified.

The S option sets the relative iterative tolerance for SSR by forming the test constant $0.05 \times 10^{-d} \times (y'y)$. Iteration ceases when two successive approximations to SSR differ in magnitude by less than this test constant. This approach in practice seems to provide at least the requested accuracy in results obtained iteratively. A specification of $d = 0$ results in the skipping of all calculations in **PART2** and thus provides an option for independent rank calculations.

The T option sets the probability level at which F tests will be conducted. When iterative calculations are necessary for hypothesis testing, the constant c of the iterative AOV algorithm [12] is selected such that the sequence of approximations to the F statistic will be monotonically decreasing. The approximation given in [20] is used to compute the probability level for the F approximation. Whenever this value falls below the level specified, iteration ceases due to lack of significance. In the event that this value stays above the level specified, iteration continues until two successive F approximations differ by less than 0.005. When the P option is specified, iteration is continued for nonsignificant F's as well, in order to also obtain final probability levels for these statistics.

The R option must be in on status to compute rank *iteratively*. Noniterative rank computations in **PART1** are not affected by this option. The terminal iterative approximation to rank need not be highly accurate since the approximations converge monotonically to an integer; furthermore, absolute rather than relative cutoff tolerances may be used. Although the preset values for these tolerances may be modified, they have been tested to obtain the correct rank with minimal iteration.

The method best suited to a relatively small number of missing cells involves computing powers of a matrix whose order is the number of missing cells. The initial matrix is formed noniteratively. Successive powers, 1, 2, 4, 8, . . . , of this matrix are formed in the array **Q**. The trace of the matrix in **Q** converges quadratically and monotonically to the rank of X_0. In this case, we cease iteration when successive traces differ by less than 0.1. The rank is taken as the next integer. The method best suited to a large number of missing cells applies the iterative AOV algorithm to dummy unit vectors for each filled cell with D replaced by Δ. The sum of the SSRs obtained for each unit vector will equal the rank of X_0. In this case, we cease iteration when two successive approximations to SSR for a given unit vector differ by less than $0.1/n_f$ where n_f is the number of filled cells. This method will be expensive timewise when n_f is large.

When the Z option is on, the following intermediate output is produced when applicable: the current E/R list; the successive traces of the matrix in array **Q** or the cumulative sum of the SSRs for unit vectors in computing rank; the approximations to SSR for the full model; the approximations to SSR for the reduced model of the hypothesis.

Although the A and C options may appear to be somewhat mundane, these options require no additional array storage and they require little additional code; furthermore, it is likely that they would be requested as part of an analysis of variance.

An option statement is written on one line as a string of selected identifiers enclosed in parentheses and preceded by an O. The identifers may be in any order. For example, with the R and Z options in off status, to specify eight significant digits of accuracy, iterative rank computations, and intermediate output, the options statement may be written as

O(R, S(8), Z)

where the commas are again cosmetic.

9. MODEL AND HYPOTHESIS STATEMENTS

Examples of the model and hypothesis statements for three factors are given in Table II along with the E/R list entries that would be produced for these statements. These examples assume that factor symbols A, B, C, and subscript symbols I, J, K were used in the factor, levels statement. The analysis-of-variance

Table II. Model and Hypothesis Statements

Statements	E/R list							
$(y_{ijkl} = \mu + a_i + b_j + c_k + e_{ijkl})$	ABC	AB	AC	A	BC	B	C	M
M + A(I) + B(J) + C(K)	0	0	0	5	0	3	2	1
H B(J)	0	0	0	5	0	−3	2	1
$(y_{ijkl} = \mu + a_i + b_{ij} + c_{ijk} + e_{ijkl})$	ABC	AB	AC	A	BC	B	C	M
M + A(I) + B(IJ) + C(IJK)	2	3	2	5	2	3	2	1
H C(IJK)	−2	3	−2	5	−2	3	−2	1
$(y_{ijkl} = \mu + a_i + b_{ij} + c_k + ac_{ik} + e_{ijkl})$	ABC	AB	AC	A	BC	B	C	M
M + A(I) + B(IJ) + C(K) + AC(IK)	0	3	6	5	0	3	2	1
H AC(IK)	0	3	−6	5	0	3	2	1
$(y_{ijkl} = \mu + a_i + b_j + ab_{ij} + c_{ijk} + e_{ijkl})$	ABC	AB	AC	A	BC	B	C	M
M + A(I) + B(J) + AB(IJ) + C(IJK)	2	7	2	5	2	3	2	1
H AB(IJ), C(IJK)	−2	−7	−2	5	−2	3	−2	1

model, as it is usually written, appears above each model statement in Table II. Notice that the variate y_{ijkl} and the error term e_{ijkl} do not appear in the model statement. The additional subscript l is also implicit; that is, the number of observations in the (i, j, k) cell depends upon the data. The rules for constructing the model statement basically parallel those used in [14] and [18]: the associated subscript of a factor must always appear along with the factor symbol in a model term; if a factor is nested, it must be nested within a factor or factors appearing to the left of it in the factor, levels statement; the ordering of the associated subscripts in the factor, levels statement must correspond with the ordering of the data. As indicated earlier, pluses, blanks, and commas serve only to improve the appearance of these statements; these characters are ignored in parsing. Should either the model or hypothesis statement require more than one line (or card), continuation to the next line is indicated by placing a slash (/) after the last model term in the current line. For example, the fourth model statement in Table II could be written as

M + A(I) + B(J) + AB(IJ)/

+ C(IJK)

As a convenience, a full factorial model may be specified as M*, an asterisk following M. Hypothesis statements are then written normally using the same symbols appearing in the factor, levels statement. The E/R list construction algorithm includes the logic necessary to detect invalid models or hypotheses incorrectly specified. During interactive use, corrections can be made when an error is detected.

Any number of models may be applied experimentally to a given set of data. This includes models with fewer factors than appear in the factor, levels statement; the data will automatically be restructured for such models.

10. FLOW OF CONTROL

Flow of control proceeds in the following manner:

(1) factor, levels statement,
(2) variable **FORMAT** statement,
(3) data,
(4) blank line (card),
(5) options statement/model statement/hypothesis statement/blank line/E.

The data are prepared with one observation per line and cell indicators preceding the observation; the variable **FORMAT** statement must reflect this order. Cell

indicators are the factor levels for the observation. There is no indicator for within cell replication and no entries are made for missing cells. A check on lexicographical ordering of indicators is included; however, this check is easily deleted if inconvenient.

Once the data have been entered, control is centered upon (5). Any number of options, model, or hypothesis statements may be entered in any order (except that a model statement must precede any hypothesis statements). Entering a blank line returns control to (1); entering the character E, for end, will terminate processing.

11. EXAMPLE

We present here a simple example, processed in batch mode, to illustrate many of the features of the global algorithm. There are three factors A, B, and C with levels of 3, 2, and 2, respectively; this information is conveyed in the initial factor, levels statement. The data are unbalanced and three cells—cells (1, 1, 2), (2, 1, 2), and (2, 2, 1)—are empty.

After the data have been read, the first option statement is entered requesting intermediate output and specifying the A, C, and R options. Summary statistics, computed immediately for the A and C options, are printed followed by the current status of the options, including default values. A model statement representing a model with three crossed factors A, B, and C and an interaction between factors A and B is entered; the corresponding E/R list is printed as intermediate output. Intermediate output of rank computations for the design matrix for the full model follows, with each iteration displaying the trace of the matrix in **Q**, until the rank of 7 has been determined. SSR for the model is next computed iteratively to approximately five significant digits with intermediate output printed for each iteration.

An hypothesis statement is entered to test the significance of factor C. The algorithm determines the rank of the design matrix for the reduced model to be six noniteratively; it also restructures the data and computes SSR noniteratively for the reduced model. The computed F statistic is found to be not significant at the 5 percent level. Another hypothesis statement is entered to test the hypothesis of no AB interaction. Again the algorithm determines rank noniteratively; however, it must iterate to determine SSR for the reduced model. Intermediate output of the monotonically increasing approximation to SSR is printed until the corresponding approximation for F falls below the 5 percent level on the second iteration.

```
F(A,B,C) L(I(3),J(2),K(2))
DATA FORMAT AND INPUT DATA-
(1X,3I2,F4.1)
 1 1 1 2.2
 1 1 1 2.8
 1 2 1 3.1
 1 2 2 4.5
 2 1 1 2.4
 2 1 1 2.7
 2 2 2 2.6
 2 2 2 5.2
 3 1 1 4.5
 3 1 2 3.6
 3 2 1 5.0
 3 2 2 3.7
 0 0 0  .0
O(Z,A,C,R)
CELL SUMS, FREQUENCIES, AND MEANS-
   CELL        SUM        FREQ.       MEAN
     1    .50000000e+01    2.     .25000000e+01
     2            (MISSING CELL)
     3    .31000000e+01    1.     .31000000e+01
     4    .45000000e+01    1.     .45000000e+01
     5    .51000000e+01    2.     .25500000e+01
     6            (MISSING CELL)
     7            (MISSING CELL)
     8    .78000000e+01    2.     .39000000e+01
     9    .45000000e+01    1.     .45000000e+01
    10    .36000000e+01    1.     .36000000e+01
    11    .50000000e+01    1.     .50000000e+01
    12    .37000000e+01    1.     .37000000e+01
```

```
CLASSIFICATION SUMS, FREQUENCIES, AND MEANS-
IJ.
     1    .50000000e+01   2.    .25000000e+01
     2    .76000000e+01   2.    .38000000e+01
     3    .51000000e+01   2.    .25500000e+01
     4    .78000000e+01   2.    .39000000e+01
     5    .81000000e+01   2.    .40500000e+01
     6    .87000000e+01   2.    .43500000e+01
I.K
     1    .81000000e+01   3.    .27000000e+01
     2    .45000000e+01   1.    .45000000e+01
     3    .51000000e+01   2.    .25500000e+01
     4    .78000000e+01   2.    .39000000e+01
     5    .95000000e+01   2.    .47500000e+01
     6    .73000000e+01   2.    .36500000e+01
I..
     1    .12600000e+02   4.    .31500000e+01
     2    .12900000e+02   4.    .32250000e+01
     3    .16800000e+02   4.    .42000000e+01
.JK
     1    .14600000e+02   5.    .29200000e+01
     2    .36000000e+01   1.    .36000000e+01
     3    .81000000e+01   2.    .40500000e+01
     4    .16000000e+02   4.    .40000000e+01
.J.
     1    .18200000e+02   6.    .30333333e+01
     2    .24100000e+02   6.    .40166667e+01
..K
     1    .22700000e+02   7.    .32428571e+01
     2    .19600000e+02   5.    .39200000e+01
...
     1    .42300000e+02  12.    .35250000e+01
OPTIONS- S= 5, T= .0500, I=100, R=1, V=0, G=0, P=0
 M+A(I)+B(J)+AB(IJ)+C(K)
E/R LIST-
    0    7    0    5    0    3    2    1
ITERATION  0, TRACE=      5.250000000
ITERATION  1, TRACE=      5.937500000
ITERATION  2, TRACE=      6.558593750
ITERATION  3, TRACE=      6.892074585
THE RANK OF THE M DESIGN MATRIX IS    7
ITERATION   1, SSRM=    .11689479e+03
ITERATION   2, SSRM=    .14589862e+03
ITERATION   3, SSRM=    .15314517e+03
ITERATION   4, SSRM=    .15495693e+03
ITERATION   5, SSRM=    .15541004e+03
ITERATION   6, SSRM=    .15552343e+03
ITERATION   7, SSRM=    .15555187e+03
ITERATION   8, SSRM=    .15555904e+03
ITERATION   9, SSRM=    .15556089e+03
ITERATION  10, SSRM=    .15556138e+03
ITERATION  11, SSRM=    .15556153e+03
ITERATION  12, SSRM=    .15556158e+03
ITERATION  12, SSR(FULL MODEL)=    .15556158e+03,
               SSE(FULL MODEL)=    .57284167e+01
 H C(K)
E/R LIST-
    0    7    0    5    0    3   -2    1
THE RANK OF THE H DESIGN MATRIX IS    6
FROM RANK COMPUTATIONS- DF(NUM)=   1, DF(DEN)=    5
ITERATION   1, SSRH=    .15545500e+03
ITERATION   1, F=         .093 , PROB(F) .GT.  .7633 VS. F LEVEL OF    .0500
SSR(REDUCED MODEL)=    .15545500e+03
 H AB(IJ)
E/R LIST-
    0   -7    0    5    0    3    2    1
THE RANK OF THE H DESIGN MATRIX IS    5
FROM RANK COMPUTATIONS- DF(NUM)=   2, DF(DEN)=    5
ITERATION   1, SSRH=    .11623292e+03
ITERATION   2, SSRH=    .14498170e+03
ITERATION   2, F=        4.617 , PROB(F) .GT.  .0734 VS. F LEVEL OF    .0500
SSR(REDUCED MODEL)=    .14498170e+03
 O(Z,V,G)
OPTIONS- S= 5, T= .0500, I=100, R=1, V=1, G=1, P=0
 M+A(I)+B(J)
THE RANK OF THE M DESIGN MATRIX IS    4
ITERATION   1, SSR(FULL MODEL)=    .15475333e+03,
               SSE(FULL MODEL)=    .65366667e+01
ESTIMATES OF EXPECTED CELL MEANS-
   CELL  ESTIMATED MEAN
     1    .26583333e+01
     2    .26583333e+01 (MISSING CELL)
     3    .36416667e+01
     4    .36416667e+01
     5    .27333333e+01
     6    .27333333e+01 (MISSING CELL)
     7    .37166667e+01 (MISSING CELL)
     8    .37166667e+01
     9    .37083333e+01
```

```
    10     .37083333e+01
    11     .46916667e+01
    12     .46916667e+01
G-INVERSE SOLUTION
A
     1   -.37500000e+00
     2   -.30000000e+00
     3    .67500000e+00
B
     1   -.49166667e+00
     2    .49166667e+00

     1    .35250000e+00
E
```

Another options statement is entered which suspends intermediate output and specifies the V and G option; again the status of the options is printed. A model statement representing the two-way classification without interaction for factors A and B is entered. All of the remaining computations are noniterative. Estimates of expected cell means are printed followed by a solution to the normal equations in response to the V and G options. Unfortunately, this example does not convey the algorithm's facility to handle large data sets and diverse models.

12. SUMMARY

We have described a comprehensive algorithm for analysis of variance which has the facility to handle large problems involving unbalanced data on small computers. At the same time, the computational efficiencies common to balanced data are realized. Array storage economy is achieved by virtue of the fact that the algorithm performs vector operations as opposed to matrix operations. Neither the design matrix of indicator variables X_0 nor the matrix $X_0'X_0$ is ever formed or stored. Ancillary computations and output amenities have been deliberately limited to conserve program storage; however, the facility for algebraic problem specifications has been included.

For some data sets and models requiring iteration, the algorithm may converge slowly so that it will be expensive timewise to achieve a high degree of accuracy; however, performance studies of the iterative AOV algorithm given in [10] and [6] indicate that the algorithm is likely to outperform matrix-oriented algorithms timewise on large problems with unbalanced data. It should always outperform these algorithms timewise when the data are balanced. Subsequent to submission of this algorithm, other authors [2], [15] have suggested using the conjugate gradient method rather than iterative improvement for the basic iterative step. The monotonicity properties of the algorithm are preserved with the conjugate gradient method, and performance should be improved whenever iteration for SSR is necessary. Subroutine step may be modified relatively easily to reflect using this method.

ACKNOWLEDGMENTS

The author wishes to acknowledge three former research assistants: Donald Piette, Clifford Pelletier, and James Petrarca. Their work in implementing the iterative AOV algorithm and extensions of this algorithm to covariance and multivariate analysis contributed to this ultimate design of the algorithm presented here. The author appreciates the efforts of Karen Hemmerle Hodge related to editing, documenting, and verifying the program.

REFERENCES

1. BARR, A.J., GOODNIGHT, J.H., SALL, J.P., AND HELWIG, J.T. A user's guide to SAS 76. SAS Institute Inc., Raleigh, N.C., 1976.
2. GOLUB, G.H., AND NASH, S.G. Non-orthogonal analysis of variance using a generalized conjugate gradient algorithm. *J. Am. Stat. Assoc. 77* (1982), 109–116.
3. HARTLEY, H.O. A plan for programming analysis of variance for general purpose computers. *Biometrics 12* (1956), 110–122.
4. HEALY, M., AND WESTMACOTT, M. Missing values in experiments analysed on automatic computers. *Appl. Stat. 5* (1956), 203–206.
5. HEMMERLE, W.J., CARNEY, E.J., JOHNSON, A.F., MENSING, R.W., AND SMITH, J. AARDVARK,

a compiler-monitor system for analysis of variance. *Reference Manual*, Iowa State Statistical Laboratory, Ames, 1963.
6. HEMMERLE, W.J., AND PIETTE, D.F. An iterative algorithm for nonorthogonal analysis of variance. In Proc. Computer Science and Statistics: 8th Annu. Symp. on the Interface (Health Sciences Computing Facility, Univ. Calif. at Los Angeles), 1975, pp. 191–195.
7. HEMMERLE, W.J., AND CARNEY, E.J. An algorithm for multivariate analysis of covariance (implemented in AARDVARK). In *Statistical Computation*, R.C. Milton and J.A. Nelder (Eds.). Academic, New York, 1969.
8. HEMMERLE, W.J. Recognizing balance with unbalanced data. *Commun. Stat. A9* (1980), 201–211.
9. HEMMERLE, W.J. Balanced hypotheses and unbalanced data. *J. Am. Stat. Assoc. 74* (1979), 794–798.
10. HEMMERLE, W.J. Extensions and improvements of recent linear model algorithms. In *Proc. ASA Statistical Computing Section* (Washington, D.C.), American Statistical Association, 1976, pp. 73–82.
11. HEMMERLE, W.J. Iterative nonorthogonal analysis of covariance. *J. Am. Stat. Assoc. 71* (1976), 195–199.
12. HEMMERLE, W.J. Nonorthogonal analysis of variance using iterative improvement and balanced residuals. *J. Am. Stat. Assoc. 69* (1974), 772–778.
13. HEMMERLE, W.J. *Statistical Computations on a Digital Computer*. Blaisdell, Waltham, Mass., 1967.
14. HEMMERLE, W.J. Algebraic specification of statistical models for analysis of variance computations. *J. ACM 11*, 2 (April 1964), 234–239.
15. MCINTOSH, A.A. Fitting linear models by conjugate gradient. Tech. Rep., Dep. Statistics, Univ. Toronto, Toronto, Ont., Canada, 1980.
16. RUBIN, D.B. A non-iterative algorithm for least squares estimation of missing values in any analysis of variance design. *Appl. Stat. 21* (1972), 136–141.
17. RYDER, B.G., AND HALL, A.D. The FORTRAN verifier: User's guide. Computing Sci. Tech. Rep. 12, Bell Telephone Laboratories, Murray Hill, N.J., 1973, Rev. 1975.
18. SCHLATER, J.E., AND HEMMERLE, W.J. Statistical computations based upon algebraically specified models. *Commun. ACM 9*, 12 (Dec. 1966), 865–869.
19. SEARLE, S.R. *Linear Models*. Wiley, New York, 1967.
20. SMILLIE, K.W., AND ANSTEY, T.H. A note on the calculation of probabilities in an F-distribution. *Commun. ACM 7*, 12 (Dec. 1964), 725.
21. SPEED, S.M., HOCKING, R.R., AND HACKNEY, O.P. Methods of analysis of linear models with unbalanced data. *J. Am. Stat. Assoc. 73* (1978), 105–112.
22. SPEED, F.M., AND HOCKING, R.R. The use of the R( ) notation with unbalanced data. *The American Statistician 30* (1976), 30–33.
23. WILKINSON, G.N. A general recursive procedure for analysis of variance. *Biometrika 57* (1970), 19–46.
24. YATES, F. The analysis of multiple classifications with unequal numbers in different classes. *J. Am. Stat. Assoc. 29* (1934), 52–66.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Distribution Service.]

```
C  **A COMPREHENSIVE, MATRIX FREE ALGORITHM FOR ANALYSIS OF VARIANCE**  MAN   1Ø
C                                                                       MAN   2Ø
C  **STATISTICS COMPUTED**                                              MAN   3Ø
C                                                                       MAN   4Ø
C    1) CELL SUMS, FREQUENCIES, AND MEANS                               MAN   5Ø
C                                                                       MAN   6Ø
C    2) CLASSIFICATION SUMS, FREQUENCIES, AND MEANS                     MAN   7Ø
C                                                                       MAN   8Ø
C    3) RANK(DESIGN MATRIX FOR FULL MODEL)                              MAN   9Ø
C                                                                       MAN  1ØØ
C    4) SSE(FULL MODEL) AND SSR(FULL MODEL)                             MAN  11Ø
C                                                                       MAN  12Ø
C    5) A G-INVERSE SOLUTION TO THE NORMAL EQUATIONS                    MAN  13Ø
C                                                                       MAN  14Ø
C    6) ESTIMATES OF EXPECTED CELL MEANS                                MAN  15Ø
C                                                                       MAN  16Ø
C    7) RANK(RESTRICTED DESIGN MATRIX FOR REDUCED MODEL)                MAN  17Ø
C                                                                       MAN  18Ø
C    8) SSE(REDUCED MODEL) AND SSR(REDUCED MODEL)                       MAN  19Ø
C                                                                       MAN  2ØØ
C    9) THE F STATISTIC FOR A SPECIFIED HYPTOTHESIS                     MAN  21Ø
```

```
C                                                                       MAN  22Ø
C   1Ø) PROBABILITY OF A GREATER F GIVEN THE HYPOTHESIS                 MAN  23Ø
C                                                                       MAN  24Ø
C  **FLOW OF CONTROL**                                                  MAN  25Ø
C                                                                       MAN  26Ø
C    1) FACTOR, LEVELS STATEMENT                                        MAN  27Ø
C                                                                       MAN  28Ø
C    2) VARIABLE FORMAT STATEMENT                                       MAN  29Ø
C                                                                       MAN  3ØØ
C    3) DATA                                                            MAN  31Ø
C                                                                       MAN  32Ø
C    4) BLANK LINE (CARD)                                               MAN  33Ø
C                                                                       MAN  34Ø
C    5) OPTIONS STATEMENT/MODEL STATEMENT/HYPOTHESIS STATEMENT/BLANK    MAN  35Ø
C       LINE/E                                                          MAN  36Ø
C                                                                       MAN  37Ø
C       E ENDS PROCESSING, BLANK LINE SENDS CONTROL TO 1), STATEMENTS   MAN  38Ø
C       RETURN CONTROL TO 5) FOLLOWING EXECUTION.                       MAN  39Ø
C                                                                       MAN  4ØØ
C  **FACTOR, LEVELS STATEMENT**                                         MAN  41Ø
C                                                                       MAN  42Ø
C    A DISTINCT ALPHABETIC CHARACTER IS USED TO NAME EACH FACTOR AND    MAN  43Ø
C    EACH SUBSCRIPT. THE STRING OF FACTOR SYMBOLS IS PLACED IN PARENS   MAN  44Ø
C    AND THEN PRECEDED BY THE LETTER F. THIS IS FOLLOWED BY THE LET-    MAN  45Ø
C    TER L AND A PARENTHESIZED LIST OF ASSOCIATED SUBSCRIPTS WITH THE   MAN  46Ø
C    NUMBER OF LEVELS FOR THE FACTOR IN PARENS FOLLOWING THE ASSOCI-    MAN  47Ø
C    ATED SUBSCRIPT.                                                    MAN  48Ø
C                                                                       MAN  49Ø
C    EXAMPLE FOR 3 FACTORS NAMED A, B, C, ASSOCIATED SUBSCRIPTS NAMED   MAN  5ØØ
C    I, J, K, AND NUMBER OF LEVELS OF THE FACTORS EQUAL TO 5, 1Ø, 15:   MAN  51Ø
C                                                                       MAN  52Ø
C      F(A,B,C) L(I(5),J(1Ø),K(15))                                     MAN  53Ø
C                                                                       MAN  54Ø
C      COMMAS AND BLANKS ARE COSMETIC                                   MAN  55Ø
C                                                                       MAN  56Ø
C      ORDERING OF THE ASSOCIATED SUBSCRIPTS IS ASSUMED TO CORRESPOND   MAN  57Ø
C      TO THE ORDERING OF THE DATA. THE NUMBER OF OBSERVATIONS WITHIN   MAN  58Ø
C      THE (I,J,K) CELL DEPENDS UPON THE DATA; NO SUBSCRIPT SYMBOL IS   MAN  59Ø
C      USED TO DENOTE WITHIN CELL REPLICATION.                          MAN  6ØØ
C                                                                       MAN  61Ø
```