Scientific Subroutines Programmer's Reference Manual

AA–1101D–TC, AD–1101D–T1

January 1982

This document describes the Scientific Subroutines Package available to users of FORTRAN RT/11 and RSX–11M, FORTRAN IV and FORTRAN 77.

This manual and its update (AD–1101D–T1) replace the *Scientific Subroutines Programmer's Reference Manual*, order number AA–1101C–TC.

OPERATING SYSTEM:	RT–11 V4.0 RSX–11M V4.0 FORTRAN IV V2.5 FORTRAN 77 V4.0
SOFTWARE:	SSP V1.3

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center. Outside the United States, orders should be directed to the nearest DIGITAL Field Sales Office or representative.

Northeast/Mid–Atlantic Region

Digital Equipment Corporation
PO Box CS2008
Nashua, New Hampshire 03061
Telephone:(603)884–6660

Central Region

Digital Equipment Corporation
Accessories and Supplies Center
1050 East Remington Road
Schaumburg, Illinois 60195
Telephone:(312)640–5612

Western Region

Digital Equipment Corporation
Accessories and Supplies Center
632 Caribbean Drive
Sunnyvale, California 94086
Telephone:(408)734–4915

digital equipment corporation • marlboro, massachusetts

First Printing, February 1975
Revised, November 1978
Revised, June 1980
Revised, October 1981
Updated, January 1982

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

CONTENTS

CONTENTS (CONT.)

CONTENTS (CONT.)

TABLES

PREFACE

MANUAL OBJECTIVES AND READER ASSUMPTIONS

The Scientific Subroutines Programmer's Reference Manual describes the Scientific Subroutines Package (SSP), a set of over 100 mathematical and statistical subroutines.

To use this manual, you should be a programmer familiar with the FORTRAN IV or FORTRAN 77 programming language and with either the RT-11 operating system (Single Job (SJ) or Foreground/Background (F/B) monitor) or with the RSX-11M or RSX-11M-PLUS operating systems.

You should also be familiar with a scientific laboratory, should understand the capabilities and operation of all instruments in your system, and should have access to instrument manufacturers' documentation.

MANUAL STRUCTURE

The Scientific Subroutines Programmer's Reference Manual contains three chapters and three appendixes.

Chapter 1 introduces the Scientific Subroutines software.

Chapter 2 describes the statistical subroutines.

Chapter 3 describes the mathematical subroutines.

Appendix A explains how to install, verify, and use the Scientific Subroutines with your own FORTRAN IV programs under the RT-11 SJ or F/B operating system.

Appendix B explains how to install, verify, and use the Scientific Subroutines with your own FORTRAN IV or FORTRAN 77 programs under the RSX-11M or RSX-11M-PLUS operating systems.

Appendix C provides an alphabetical list of the Scientific Subroutines.

RELATED DOCUMENTS

The following documents provide more information about the RT-11, RSX-11M, or RSX-11M-PLUS operating systems and the FORTRAN IV and FORTRAN 77 programming languages.

Reference	Order Number
PDP-11 FORTRAN 77 User's Guide	AA-1884D-TC
IAS/RSX-11 FORTRAN IV User's Guide	AA-1936E-TC
PDP-11 FORTRAN Language Reference Manual	AA-1855D-TC
RSX-11M/RSX-11S Information Directory and Index	AA-2593F-TC
Introduction to RSX-11M and RSX-11M-PLUS	AA-L763A-TC
RSX-11M/M-PLUS MCR Operations Manual	AA-L678A-TC
RSX-11M/M-PLUS Task Builder Manual	AA-L680A-TC
RSX-11M/M-PLUS Utilities Manual	AA-L681A-TC
Introduction to RT-11	AA-5281B-TC
RT-11 Programmer's Reference Manual	AA-H378A-TC
RT-11/RSTS/E FORTRAN IV User's Guide	AA-5749B-TC
RT-11 Software Support Manual	AA-H379A-TC
RT-11 System User's Guide	AA-5279B-TC
RT-11 System Message Manual	AA-5284C-TC

Bibliographies of non-DIGITAL technical publications appear at the end of Chapters 2 and 3. Many subroutine descriptions in Chapters 2 and 3 contain brief references to authors and publications in these bibliographies. For complete information about a publication referenced in a subroutine description, consult the bibliography at the end of the chapter where the reference appears.

DOCUMENTATION CONVENTIONS

The following conventions apply to this manual:

- In programming examples, all information the computer prints appears in black. All commands and responses you type appear in red.
- (RET) means you must press the RETURN key on your terminal.
- You produce certain characters by typing a combination of keys together. For example, hold down the CTRL key and type the letter C to produce the CTRL C character. Combinations such as this are documented as (CTRL/C).

- Many commands in this manual contain the expression dvn:. When you execute the commands, specify a device and unit number in place of dvn:. If you do not include a unit number, the system uses unit 0 as a default.

  For a list of devices and their abbreviations, see the RT-11 System User's Guide or the RSX-11M/M-PLUS MCR Operations Manual.

- In examples of commands or file names, capital letters represent actual commands, file names or file types. You must type these exactly as they appear. Lower case letters mean that you must supply a name.

- The term RSX-11M/M-PLUS means either or both the RSX-11M and RSX-11M-PLUS operating systems.

- Brackets represent optional elements in a command. When you use an option, do not type the brackets in the command line.

NOTE

Under RSX-11M/RSX-11M-PLUS, brackets are also a part of the User File Directory (UFD) portion of file specifications, that is, [group,member]. When you type this portion of a file specification, brackets are required syntax elements. You must type the brackets in the command line.

CHAPTER 1

INTRODUCTION

The Scientific Subroutines Programmer's Reference Manual accompanies the Scientific Subroutines Package (SSP), a set of over 100 mathematical and statistical subroutines.

This manual describes the subroutines and explains how to use them with your FORTRAN programs. The manual contains three chapters and three appendixes. This first chapter is an introduction to the package. It provides an overview of the subroutines and of the manual. Chapters 2 and 3 contain descriptions of the Scientific Subroutines. Each description defines all arguments and outlines the method the subroutine uses to perform its function. Descriptions of the more complex subroutines in Chapters 2 and 3 include bibliographical references. The bibliographies are at the end of Chapters 2 and 3. Each bibliography gives complete information about publications cited in the subroutine descriptions.

Each description also names any test program that calls the subroutine. The test program is named as follows:

(Test: prog.ext)

where: prog is the name of the test program.

.ext is the file extension.

.ext=.FOR for the RT-11 operating system.

.ext=.FTN for the RSX-11M operating system.

Appendixes A and B explain how to install, verify, and use the Scientific Subroutines under the RT-11 and RSX-11M/M-PLUS operating systems respectively. Appendix C lists the Scientific Subroutines alphabetically.

1.1 THE SCIENTIFIC SUBROUTINES PACKAGE

The Scientific Subroutines Package consists of over 100 subroutines that you can call from any FORTRAN IV program running under the RT-11 operating system and from any FORTRAN IV or FORTRAN 77 program running under the RSX-11M/M-PLUS operating systems.

NOTE

FORTRAN 77 V4.0 is the next version of FORTRAN IV-PLUS V3.0. FORTRAN 77 is so named because it adheres to the 1977 ANSI subset standard for FORTRAN programming languages. Because FORTRAN 77 is compatible with FORTRAN IV-PLUS V3.0, your FORTRAN IV-PLUS V3.0 programs can run under FORTRAN 77.

Each subroutine in the package performs a different statistical or mathematical operation commonly required in scientific programming. Many of the larger statistical subroutines, however, are programmed as collections of smaller routines to let you use them more easily in large, overlaid programs.

None of the subroutines needs input/output operations. All of the subroutines use single-precision values.

Many of the subroutines, however, can use double-precision values in calculations (at a loss of speed and an increase in memory). The individual subroutine source file contains the instructions for making the necessary modifications to convert from single-precision to double-precision calculations.

1.1.1 Subroutine Arrays

To save as much memory space as possible, the subroutines use only dimension statements to access an array. They do not use them as an upper limitation on array size. The subroutines always treat arrays as vectors and calculate subscripts when required.

Three types of arrays are used in scientific programming. They are, in decreasing order of occurrence:

1. General matrices including vectors (storage mode 0).
2. Symmetric matrices (storage mode 1).
3. Diagonal matrices (storage mode 2).

Symmetric and diagonal matrices require less memory than general matrices. Where a general matrix of the order N by N requires N*N storage locations, a symmetric matrix of the same order uses N*(N+1)/2 locations, and a diagonal matrix requires only N locations.

The subroutine LOC calculates most of the matrix references in the Scientific Subroutines. LOC determines the relative position of any element in a vector array. LOC is supplied as a FORTRAN subroutine. To increase execution speed, you could write an assembly-language version of LOC.

1.1.2 The Scientific Subroutines Package Distribution Kit

The distribution kit for the Scientific Subroutines Package consists of up to two mass-storage volumes containing the Scientific Subroutines Package, this manual, a software product description (SPD), and other forms.

The distribution volume contains the following kinds of files:

1. The Scientific Subroutines files. These are FORTRAN source files.

2. Test program files. These are also FORTRAN source files. Each contains a FORTRAN program that calls one or more of the Scientific Subroutines.

3. Test program data files. These are required by some of the test programs.

4. Indirect-command files. These files compile the test programs then link or task-build them and run them.

For a complete table of files in the SSP distribution kit, see Section A.3.1 if you are using RT-11, or Section B.3.1 if you are using RSX-11M/M-PLUS.

CHAPTER 2

STATISTICAL SUBROUTINES

2.1 DATA SCREENING SUBROUTINES

The following sections describe the data screening subroutines.

2.1.1 ABSNT Subroutine

ABSNT tests for missing (or zero) values for each observation in a general matrix A (Test: STAT.ext).

Usage: CALL ABSNT (A,S,NO,NV)

Parameter	Description
A	Observation matrix, NO by NV.
S	Output vector of length NO indicating the following codes for each observation: 1 There is no missing or zero value. 0 At least one value is missing or zero.
NO	Number of observations. NO must be greater than or equal to 1.
NV	Number of variables for each observation. NV must be greater than or equal to 1.

Method:

ABSNT tests each row (observation) of the matrix A. If it does not find a missing or zero value, it places a one in S(J). If it does find at least one missing or zero value, it places a zero in S(J).

2.1.2 BOUND Subroutine

BOUND selects from a set (or a subset) of observations, the number of observations under, between, and over two given bounds for each variable (Test: STAT.ext).

Usage: CALL BOUND (A,S,BLO,BHI,UNDER,BETW,OVER,NO,NV,IER)

Parameter	Description
A	Observation matrix, NO by NV.
S	Vector indicating subset of A. Only those observations with a nonzero S(J) are considered. The vector length is NO.
BLO	Input vector of lower bounds on all variables. The vector length is NV.
BHI	Input vector of upper bounds on all variables. The vector length is NV.
UNDER	Output vector indicating, for each variable, the number of observations under lower bounds. The vector length is NV.
BETW	Output vector indicating, for each variable, the number of observations equal to or between the lower and upper bounds. Vector length is NV.
OVER	Output vector indicating, for each variable, the number of observations over upper bounds. Vector length is NV.
NO	Number of observations.
NV	Number of variables for each observation.
IER	Resultant error code: 0 No error. 1 S is null. VMIN=1.E37, SD=AVER=0. 2 S has only one nonzero element. VMIN=VMAX, SD=0.0.

Method:

BOUND tests each row (observation) of matrix A with a corresponding nonzero element in the vector S. It compares observations with specified lower and upper variable bounds and keeps a count in vectors UNDER, BETW, and OVER.

2.1.3 SUBMX Subroutine

SUBMX builds a subset matrix. Based on vector S derived from subroutine SUBST or ABSNT, SUBMX copies from a larger matrix of observation data a subset matrix of those observations which have satisfied a certain condition. This subroutine is normally used before the statistical analyses (multiple regression or factor analysis) (Test: STAT.ext).

Usage: CALL SUBMX (A,D,S,NO,NV,N)

Parameter	Description
A	Input matrix of observations, NO by NV.
D	Output matrix of observations N by NV.

Parameter	Description
S	Input vector of length NO containing the codes derived from subroutine SUBST or ABSNT.
NO	Number of observations. NO must be greater than or equal to 1.
NV	Number of variables.
N	Output variable containing the number of nonzero codes in vector S.

Remarks:

Matrix D can be in the same location as matrix A.

Method:

SUBMX determines whether S(I) contains a nonzero code. If it does, the Ith observation is copied from the input matrix to the output matrix.

2.1.4 SUBST Subroutine

SUBST derives a subset vector indicating which observations in a set have satisfied certain conditions on the variables (Test: DASCR.ext).

Usage: CALL SUBST (A,C,R,B,S,NO,NV,NC)

Parameter	Description
A	Observation matrix, NO by NV.
C	Input matrix, 3 by NC, of conditions to be considered. The first element of each column of C represents the number of the variable (column of the matrix A) to be tested. The second element of each column is one of the following relational codes:

Code	Element
1.	For LT (less than)
2.	For LE (less than or equal to)
3.	For EQ (equal to)
4.	For NE (not equal to)
5.	For GE (greater than or equal to)
6.	For GT (greater than)

The third element of each column is a quantity to be used for comparison with the observation values. For example, the following in column C:

```
 2.
 5.
92.5
```

causes the second variable to be tested for greater than or equal to 92.5.

Parameter	Description
R	Working vector used to store intermediate results of above tests on a single observation. If condition is satisfied, R(I) is set to 1. If it is not, R(I) is set to 0. Vector length is NC.

B	Name of subroutine you must supply. The subroutine must be defined by an external statement in the calling program. It consists of a Boolean expression linking the intermediate values stored in vector R. The Boolean operators are * for AND and + for OR. For example:
	
	`SUBROUTINE BOOL (R,T)`
	`DIMENSION R(3)`
	`T=R(1)*(R(2)+R(3))`
	`RETURN`
	`END`
	
	The above expression is tested for: R(1).AND.(R(2).OR.R(3))

S	Output vector indicating, for each observation, whether or not proposition B is satisfied. If it is, S(I) is zero. The vector length is NO.
NO	Number of observations. NO must be greater than or equal to 1.
NV	Number of variables. NV must be greater than or equal to 1.
NC	Number of basic conditions to be satisfied. NC must be greater than or equal to 1.

Subroutines and Function Subprograms Required:

B - The name of the subroutine you must supply can be different, (for example BOOL) but the subroutine SUBST always calls it as B. For subroutine SUBST to do this, you must define the name of the subroutine by an external statement in the calling program. You must also list the name in the CALL SUBST statement. (See Usage above.)

Method:

SUBST performs the following for each observation:

1. SUBST analyzes the condition matrix to determine which variables are to be examined.

2. SUBST forms immediate vector R.

3. SUBST evaluates the Boolean expression (in subroutine B) to derive the element in subset vector S corresponding to the observation.

2.1.5 TAB1 Subroutine

TAB1 tabulates for one variable in an observation matrix (or a matrix subset) the frequency and percent over given class intervals. In addition, TAB1 calculates for the same variable the total, mean, standard deviation, minimum, and maximum (Test: DASCR.ext).

Usage: CALL TAB1(A,S,NOVAR,UBO,FREQ,PCT,STATS,NO,NV)

Parameter	Description
A	Observation matrix, NO by NV.
S	Input vector giving subset of A. Only those observations with a corresponding nonzero S(J) are considered. Vector length is NO.
NOVAR	The variable to be tabulated.
UBO	Input vector giving lower limit, number of intervals, and upper limit of variable to be tabulated in UBO(1), UBO(2), and UBO(3) respectively. If lower limit is equal to upper limit, the program uses the minimum and maximum values of the variable. Number of intervals, UBO(2), must include two cells for values under and above limits. Vector length is 3.
FREQ	Output vector of frequencies. Vector length is UBO(2).
PCT	Output vector of relative frequencies. Vector length is UBO(2).
STATS	Output vector of summary statistics, that is, total, mean, standard deviation, minimum and maximum. Vector length is 5. If S is null, the total, average, and standard deviation equal 0, minimum equals -1.E75, and maximum equals 1.E75.
NO	Number of observations. NO must be greater than or equal to 1.
NV	Number of variables for each observation. NV must be greater than or equal to 1.

Method:

TAB1 calculates the interval size from the given information or optionally, from the minimum and maximum values for variable NOVAR. TAB1 then calculates the frequencies, percent frequencies, and summary statistics. The divisor for standard deviation is one less than the number of observations used.

2.1.6 TAB2 Subroutine

TAB2 performs a 2-way classification for two variables in an observation matrix (or a matrix subset) of the frequency, percent frequency, and other statistics over given class intervals (Test: STAT.ext).

Usage: CALL TAB2(A,S,NOV,UBO,FREQ,PCT,STAT1,STAT2,NO,NV)

Parameter	Description
A	Observation matrix, NO by NV.
S	Input vector giving subset of A. Only those observations with a corresponding nonzero S(J) are considered. Vector length is NO.
NOV	Variables to be cross-tabulated. NOV(1) is variable 1, NOV(2) is variable 2. Vector length is 2. NOV must be greater than or equal to 1 and less than or equal to NV.
UBO	3 by 2 matrix giving lower limit, number of intervals, and upper limit of both variables to be tabulated (first column for variable 1, second column for variable 2). If lower limit is equal to upper limit for variable 1, the program uses the minimum and maximum values on each variable. Number of intervals must include two cells for under and above limits.
FREQ	Output matrix of frequencies in the 2-way classification. Order of matrix is INT1 by INT2, where INT1 is the number of intervals of variable 1 and INT2 is the number of intervals of variable 2. INT1 and INT2 must be specified in the second position of respective column of UBO matrix.
PCT	Output matrix of percent frequencies, same order as FREQ.
STAT1	Output matrix summarizing totals, means, and standard deviations for each class interval of variable 1. Order of matrix is 3 by INT1.
STAT2	Same as STAT1 but over variable 2. Order of matrix is 3 by INT2.
NO	Number of observations. NO must be greater than or equal to 1.
NV	Number of variables for each observation. NV must be greater than or equal to 1.

Remarks:

If S is null, the output areas are set to zero.

Method:

TAB2 calculates interval sizes for both variables from the given information or, optionally, from the minimum and maximum values. TAB2 also develops the frequency and percent frequency matrices. Then, it calculates matrices STAT1 and STAT2 summarizing totals, means, and standard deviations.

The divisor for standard deviation is one less than the number of observations used in each class interval.

2.1.7 TALLY Subroutine

TALLY calculates the total, the mean, the standard deviation, the minimum, and the maximum for each variable in a set (or a subset) of observations (Test: STAT.ext).

Usage: CALL TALLY(A,S,TOTAL,AVER,SD,VMIN,VMAX,NO,NV,IER)

Parameter	Description
A	Observation matrix, NO by NV.
S	Input vector indicating subset of A. Only those observations with a nonzero S(J) are considered. Vector length is NO.
TOTAL	Output vector of totals of each variable. Vector length is NV.
AVER	Output vector of means of each variable. Vector length is NV.
SD	Output vector of standard deviations of each variable. Vector length is NV.
VMIN	Output vector of minima of each variable. Vector length is NV.
VMAX	Output vector of maxima of each variable. Vector length is NV.
NO	Number of observations.
NV	Number of variables for each observation.
IER	Resultant error code: 0 No error. 1 S is null. VMIN=1.E37, SD=AVER=0. 2 S has only one nonzero element. VMIN=VMAX, SD=0.0.

Method:

TALLY analyzes all observations corresponding to a nonzero element in vector S for each variable in matrix A. Then, it accumulates totals and finds minimum and maximum values. Following this, TALLY calculates mean and standard deviations.

The divisor for standard deviation is one less than the number of observations used.

2.2 ELEMENTARY STATISTICS SUBROUTINES

The sections that follow describe the elementary statistics subroutines.

2.2.1 MOMEN Subroutine

MOMEN finds the first four moments for grouped data on equal class intervals (Test: NPAR2.ext).

Usage: CALL MOMEN(F,UBO,NOP,ANS)

Parameter	Description
F	Grouped data (frequencies). Given as a vector of length (UBO(3)-UBO(1))/UBO(2).
UBO	Three-cell vector, UBO(1) is a lower bound and UBO(3) is an upper bound on data. UBO(2) is a class interval. Note that UBO(3) must be greater than UBO(1).
NOP	Option parameter. If NOP=1, ANS(1)=mean. If NOP=2, ANS(2)=second moment. If NOP=3, ANS(3)=third moment. If NOP=4, ANS(4)=fourth moment. If NOP=5, all four moments are filled in.
ANS	Output vector of length 4 into which moments are put.

Remarks:

The first moment is not central but the value of the mean itself. The mean is always calculated. Moments are biased and not corrected for grouping.

Method:

Refer to Kendall 1977.

2.2.2 TTSTT Subroutine

TTSTT finds certain T-statistics on the means of populations (Test: STAT.ext).

Usage: CALL TTSTT(A,NA,B,NB,NOP,NDF,ANS)

Parameter	Description
A	Input vector of length NA containing data.
NA	Number of observations in A.
B	Input vector of length NB containing data.
NB	Number of observations in B.
NOP	Options for various hypotheses: 1 That population mean of B equals given value A (set NA=1). 2 That population mean of B equals population mean of A, given that the variance of B equals the variance of A.

Parameter	Description
	3 That population mean of B equals population mean of A, given that the variance of B is not equal to the variance of A. 4 That population mean of B equals population mean of A, given no information about the variances of A and B (set NA=NB).
NDF	Output variable containing degrees of freedom associated with the T-statistic calculated.
ANS	T-statistic for given hypothesis.

Remarks:

NA and NB must be greater than 1, except that NA=1 in option 1. NA and NB must be the same in option 4. If NOP is other than 1, 2, 3, or 4, degrees of freedom and T-statistic are not calculated. NDF and ANS are set to zero.

Method:

Refer to Ostel 1975.

2.3 CORRELATION SUBROUTINE

The section that follows describes the correlation subroutine.

2.3.1 CORRE Subroutine

CORRE computes means, standard deviations, sums of cross-products of deviations, and correlation coefficients (Tests: FACTO.ext, MCANO.ext).

Usage: CALL CORRE(N,M,IO,X,XBAR,STD,RX,R,B,D,T)

Parameter	Description
N	Number of observations. N must be greater than or equal to 2.
M	Number of variables. M must be greater than or equal to 1.
IO	Option code for input data. 0 If data is to be read in from input device in the special subroutine named DATA. (See subroutines used by this subroutine below.) 1 If all data is already in memory.
X	If IO=0, the value of X is 0.0. If IO=1, X is the input matrix (N by M) containing data.
XBAR	Output vector of length M containing means.

Parameter	Description
STD	Output vector of length M containing standard deviations.
RX	Output matrix (M by M) containing sums of cross-products of deviations from means.
R	Output matrix (only upper triangular portion of the symmetric matrix of M by M) containing correlation coefficients (storage mode of 1).
B	Output vector of length M containing the diagonal of the matrix of sums of cross-products of deviations from means.
D	Working vector of length M.
T	Working vector of length M.

Subroutines and Function Subprograms Required:

DATA(M,D) - You must provide this subroutine.

1. If IO=0, this subroutine is expected to furnish an observation in vector D from an external input device.

2. If IO=1, this subroutine is not used by CORRE but must be provided when building the executable file. If you have not supplied a subroutine named DATA, the following is suggested:

```
SUBROUTINE DATA
RETURN
END
```

Method:

CORRE computes product-moment correlation coefficients.

2.4 MULTIPLE LINEAR REGRESSION SUBROUTINES

The sections that follow describe the multiple linear regression subroutines.

2.4.1 MULTR Subroutine

MULTR performs a multiple linear regression analysis for a dependent variable and a set of independent variables. This subroutine normally performs multiple and polynomial regression analyses (Test: POLRG.ext).

Usage: CALL MULTR(N,K,XBAR,STD,D,RX,RY,ISAVE,B,SB,T,ANS)

Parameter	Description
N	Number of observations.
K	Number of independent variables in this regression.

Parameter	Description
XBAR	Input vector of length M containing means of all variables. M is the number of variables.
STD	Input vector of length M containing standard deviations of all variables.
D	Input vector of length M containing the diagonal of the matrix of sums of cross products of deviations from means for all vectors.
RX	Input matrix (K by K) containing the inverse of intercorrelations among independent variables.
RY	Input vector of length K containing intercorrelations of independent variables with a dependent variable.
ISAVE	Input vector of length K+1 containing subscripts of independent variables in ascending order. The subscript of the dependent variable is stored in the last position, that is, the K+1 position.
B	Output vector of length K containing regression coefficients.
SB	Output vector of length K containing standard deviations of regression coefficients.
T	Output vector of length K containing T-values.
ANS	Output vector of length 10 containing the following information:

ANS(1) Intercept
ANS(2) Multiple correlation coefficient
ANS(3) Standard error of estimate
ANS(4) Sum of squares due to regression (SSR)
ANS(5) Degrees of freedom associated with SSR
ANS(6) Mean square of SSR
ANS(7) Residual sum of squares (RSS)
ANS(8) Degrees of freedom associated with RSS
ANS(9) Mean square of RSS
ANS(10) F-value

Remarks:

N must be greater than K+1.

Method:

MULTR uses the Gauss-Jordan method to solve normal equations. Refer to Cooley and Lohnes 1962, and Ostel 1975.

2.4.2 ORDER Subroutine

ORDER constructs a subset matrix of intercorrelations among independent variables and a vector of intercorrelations of independent variables with a dependent variable from a larger matrix of correlation coefficients. This subroutine is normally used in the performance of multiple and polynomial regression analyses (Test: POLRG.ext).

Usage: CALL ORDER(M,R,NDEP,K,ISAVE,RX,RY)

Parameter	Description
M	Number of variables and order of matrix R.
R	Input matrix containing correlation coefficients. This subroutine expects only the upper triangular portion of the symmetric matrix to be stored (by column) in R (storage mode of 1).
NDEP	The subscript number of the dependent variable.
K	Number of independent variables to be included in the forthcoming regression. K must be greater than or equal to 1.
ISAVE	Input vector of length K+1 containing, in ascending order, the subscript numbers of K independent variables to be included in the forthcoming regression. Upon returning to the calling routine, this vector contains, in addition, the subscript number of the dependent variable in position K+1.
RX	Output matrix (K by K) containing intercorrelations among independent variables to be used in forthcoming regression.
RY	Output vector of length K containing intercorrelations of independent variables with dependent variables.

Method:

ORDER constructs the matrix RX and the vector RY from the subscript numbers of the variables to be included in the forthcoming regression.

2.5 POLYNOMIAL REGRESSION SUBROUTINE

The section that follows describes the polynomial regression subroutine.

2.5.1 GDATA Subroutine

GDATA generates independent variables up to the Mth power (the highest degree polynomial specified) and computes means, standard deviations, and correlation coefficients. This subroutine is normally called before subroutines ORDER, MINV, and MULTR in the performance of a polynomial regression (Test: POLRG.ext).

Usage: CALL GDATA(N,M,X,XBAR,STD,D,SUMSQ)

Parameter	Description
N	Number of observations.
M	The highest degree polynomial to be fitted.

Parameter	Description
X	Input matrix (N by M+1). When GDATA is called, data for the independent variable is stored in the first column of matrix X, and data for the dependent variable is stored in the last column of the matrix. Upon returning to the calling routine, generated powers of the independent variable are stored in columns 2 through M.
XBAR	Output vector of length M+1 containing means of independent and dependent variables.
STD	Output vector of length M+1 containing standard deviations of independent and dependent variables.
D	Output matrix (only upper triangular portion of the symmetric matrix of M+1 by M+1) containing correlation coefficients (storage mode of 1).
SUMSQ	Output vector of length M+1 containing sums of products of deviations from means of independent and dependent variables.

Remarks:

N must be greater than M+1. If M is equal to 5 or greater, single precision may not be sufficient to give satisfactory computational results.

Method:

Refer to Ostel 1975.

2.6 CANONICAL CORRELATION SUBROUTINES

The sections that follow describe the canonical correlations subroutines.

2.6.1 CANOR Subroutine

CANOR calculates the canonical correlations between two sets of variables. CANOR is normally preceded by a call to subroutine CORRE (Test: MCANO.ext).

Usage: CALL CANOR(N,MP,MQ,RR,ROOTS,WLAM,CANR,CHISQ,NDF,COEFR, COEFL,R)

Parameter	Description
N	Number of observations.
MP	Number of left-hand variables.
MQ	Number of right-hand variables.
RR	Input matrix (only upper triangular portion of the symmetric matrix of M by M, where M=MP+MQ) containing correlation coefficients (storage mode of 1).

Parameter	Description
ROOTS	Output vector of length MQ containing eigenvalues computed in the NROOT subroutine.
WLAM	Output vector of length MQ containing lambda.
CANR	Output vector of length MQ containing canonical correlations.
CHISQ	Output vector of length MQ containing the values of chi-squares.
NDF	Output vector of length MQ containing the degrees of freedom associated with chi-squares.
COEFR	Output matrix (MQ by MQ) containing MQ sets of right-hand coefficients column-wise.
COEFL	Output matrix (MQ by MP) containing MQ sets of left-hand coefficients column-wise.
R	Work matrix (M by M).

Remarks:

The number of left-hand variables (MP) should be greater than or equal to the number of right-hand variables (MQ). The values of canonical correlation, lambda, chi-square, degrees of freedom, and canonical coefficients are computed only for those eigenvalues in roots that are greater than zero.

Method:

Refer to Cooley and Lohnes 1962.

2.6.2 NROOT Subroutine

NROOT computes the eigenvalues and eigenvectors of a real nonsymmetric matrix of the form B-inverse times A. This subroutine is normally called by subroutine CANOR in performing a canonical correlation analysis (Test: MCANO.ext).

Usage: CALL NROOT(M,A,B,XL,X)

Parameter	Description
M	Order of square matrices A, B, and X.
A	Input matrix (M by M).
B	Input matrix (M by M).
XL	Output vector of length M containing eigenvalues of B-inverse times A.
X	Output matrix (M by M) containing eigenvectors column-wise.

Subroutines and Function Subprograms Required:

EIGEN

Method:

Refer to Cooley and Lohnes 1962.

2.7 ANALYSIS OF VARIANCE SUBROUTINES

The sections that follow describe the analysis of variance subroutines.

2.7.1 AVCAL Subroutine

AVCAL performs the calculus of a factorial experiment using operator sigma and operator delta. AVCAL is preceded by subroutine AVDAT and followed by subroutine MEANQ in analyzing variance for a complete factorial design (Test: ANOVA.ext).

Usage: CALL AVCAL(K,LEVEL,X,L,ISTEP,LASTS)

Parameter	Description
K	Number of variables (factors). K must be greater than 1.
LEVEL	Input vector of length K containing levels (categories) within each variable.
X	Input vector containing data. Data has been placed in vector X by subroutine AVDAT. The length of X is (LEVEL(1)+1)*(LEVEL(2)+1)*...*(LEVEL(K)+1).
L	The position in vector X where the last input data is located. L has been calculated by subroutine AVDAT.
ISTEP	Input vector of length K containing storage control steps which have been calculated by subroutine AVDAT.
LASTS	Working vector of length K.

Remarks:

This subroutine must follow subroutine AVDAT.

Method:

The method is based on the technique discussed by H.O. Hartley. (Ralston and Wilf, eds. 1962).

2.7.2 AVDAT Subroutine

AVDAT places data for variance analysis in properly distributed positions of storage. This subroutine is normally followed by calls to AVCAL and MEANQ subroutines analyzing variance for a complete factorial design (Test: ANOVA.ext).

Usage: CALL AVDAT(K,LEVEL,N,X,L,ISTEP,KOUNT)

Parameter	Description
K	Number of variables (factors). K must be greater than 1.
LEVEL	Input vector of length K containing levels (categories) within each variable.
N	Total number of data points read.
X	When AVDAT is called, this vector contains data in locations X(1) through X(N). Upon returning to the calling routine, the vector contains the data in properly redistributed locations of vector X. The length of vector X is calculated by: (1) adding 1 to each level of variable and (2) obtaining the cumulative product of all levels. The length of X equals (LEVEL(1)+1)*(LEVEL(2)+1)*...*(LEVEL(K)+1).
L	Output variable containing the position in vector X where the last input data is stored.
ISTEP	Output vector of length K containing control steps which are used to locate data in proper positions of vector X.
KOUNT	Working vector of length K.

Remarks:

Input data must be arranged in the following manner. Consider the 3-way analysis of variance design, where one variable has three levels and the other two variables have two levels. The data may be represented in the form: X(I,J,K), I=1,2,3, J=1,2, K=1,2. In arranging data, the subscript, I, changes first. When I=3, the next subscript, J, changes and so on until I=3, J=2, and K=2.

Method:

The method is based on the technique discussed by H. O. Hartley. (Ralston and Wilf, eds. 1962).

2.7.3 MEANQ Subroutine

MEANQ computes sum of squares, degrees of freedom, and mean square using the mean square operator. MEANQ normally follows calls to AVDAT and AVCAL subroutines in analyzing variance for a complete factorial design (Test: ANOVA.ext).

Usage: CALL MEANQ(K,LEVEL,X,GMEAN,SUMSQ,NDF,SMEAN,MSTEP,KOUNT, LASTS)

Parameter	Description
K	Number of variables (factors). K must be greater than 1.
LEVEL	Input vector of length K containing levels (categories) within each variable.

Parameter	Description
X	Input vector containing the result of the sigma and delta operators. The length of X is: (LEVEL(1)+1)*(LEVEL(2)+1)*...*(LEVEL(K)+1).
GMEAN	Output variable containing grand mean.
SUMSQ	Output vector containing sums of squares. The length of SUMSQ is 2 to the Kth power minus 1, (2**K)-1.
NDF	Output vector containing degrees of freedom. The length of NDF is 2 to the Kth power minus 1, (2**K)-1.
SMEAN	Output vector containing mean squares. The length of SMEAN is 2 to the Kth power minus 1, (2**K)-1.
MSTEP	Working vector of length K.
KOUNT	Working vector of length K.
LASTS	Working vector of length K.

Remarks:

This subroutine must follow subroutine AVCAL.

Method:

The method is based on the technique discussed by H. O. Hartley. (Ralston and Wilf, eds. 1962).

2.8 DISCRIMINANT ANALYSIS SUBROUTINES

The sections that follow describe the discriminant analysis subroutines.

2.8.1 DISCR Subroutine

DISCR computes a set of linear functions that are indexes for classifying an individual into one of several groups. Normally this subroutine is used in the performance of discriminant analysis (Test: MDISC.ext).

Usage: CALL DISCR(K,M,N,X,XBAR,D,CMEAN,V,C,P,LG)

Parameter	Description
K	Number of groups. K must be greater than 1.
M	Number of variables.
N	Input vector of length K containing sample sizes of groups.

Parameter	Description
X	Input vector containing data in the manner equivalent to 3-dimensional FORTRAN array, X(1,1,1), X(2,1,1), X(3,1,1) and so forth. The first subscript is the case number, the second is the variable number, and the third is the group number. The length of vector X is equal to the total number of data points, T*M, where T=N(1)+N(2)+...+N(K).
XBAR	Input matrix (M by K) containing means of M variables in K groups.
D	Input matrix (M by M) containing the inverse of pooled dispersion matrix.
CMEAN	Output vector of length M containing common means.
V	Output variable containing generalized Mahalanobis D-square.
C	Output matrix ((M+1) by K) containing the coefficients of discriminant functions. The first position of each column (function) contains the value of the constant for that function.
P	Output vector containing the probability associated with the largest discriminant function of all cases in all groups. Calculated results are stored in a manner equivalent to a two-dimensional area (the first subscript is the case number, the second is the group number). Vector P has a length equal to the total number of cases, T (T=N(1)+N(2)+...+N(K)).
LG	Output vector containing the subscripts of the largest discriminant functions stored in vector P. The length of vector LG is the same as the length of vector P.

Remarks:

The number of variables must be greater than or equal to the number of groups.

Method:

Refer to Dixon, ed. 1977 and Anderson 1958.

2.8.2 DMATX Subroutine

DMATX computes means of variables in each group and a pooled dispersion matrix for all the groups. Normally PMATX performs discriminant analysis (Test: MDISC.ext).

Usage: CALL DMATX(K,M,N,X,XBAR,D,CMEAN)

Parameter	Description
K	Number of groups.
M	Number of variables (must be the same for all groups).
N	Input vector of length K containing sample sizes of groups.
X	Input vector containing data in the manner equivalent to a 3-dimensional FORTRAN array, X(1,1,1), X(2,1,1), X(3,1,1), and so forth. The first subscript is the case number, the second is the variable number, and the third is the group number. The length of vector X is equal to the total number of data points T*M, where T=N(1)+N(2)+...+N(K).
XBAR	Output matrix (M by K) containing means of variables in K groups.
D	Output matrix (M by M) containing pooled dispersion.
CMEAN	Working vector of length M.

Remarks:

The number of variables must be greater than or equal to the number of groups.

Method:

Refer to Dixon, ed. 1977 and Anderson 1958.

2.9 FACTOR ANALYSIS SUBROUTINES

The sections that follow describe the factor analysis subroutines.

2.9.1 LOAD Subroutine

LOAD computes a factor matrix (loading) from eigenvalues and associated eigenvectors. This subroutine normally occurs in a sequence of calls to subroutines CORRE, EIGEN, TRACE, LOAD, and VARMX in the performance of a factor analysis (Test: FACTO.ext).

Usage: CALL LOAD(M,K,R,V)

Parameter	Description
M	Number of variables.
K	Number of factors. K must be greater than or equal to 1 and less than or equal to M.
R	A matrix (symmetric and stored in compressed form with only upper triangle by column in memory) containing eigenvalues in diagonal. Eigenvalues are arranged in descending order, and first K eigenvalues are used by this subroutine. The order of matrix R is M by M. Only M*(M+1)/2 elements are in storage (storage mode of 1).

Parameter	Description
V	When LOAD is called, matrix V (M by M) contains eigenvectors column-wise. Upon returning to the calling program, matrix V contains a factor matrix (M by K).

Method:

LOAD converts normalized eigenvectors to the factor pattern by multiplying the elements of each vector by the square root of the corresponding eigenvalue.

2.9.2 TRACE Subroutine

TRACE computes the cumulative percentage of eigenvalues greater than or equal to a specified constant. TRACE normally occurs in a sequence of calls to subroutines CORRE, EIGEN, TRACE, LOAD, and VARMX in the performance of a factor analysis (Test: FACTO.ext).

Usage: CALL TRACE(M,R,CON,K,D)

Parameter	Description
M	Number of variables.
R	Input matrix (symmetric and stored in compressed form with only upper triangle by column in memory) containing eigenvalues in diagonal. Eigenvalues are arranged in descending order. The order of matrix R is M by M. Only M*(M+1)/2 elements are in storage (storage mode of 1).
CON	A constant used to decide how many eigenvalues to retain.
K	Output variable containing the number of eigenvalues greater than or equal to CON. (K is the number of factors.)
D	Output vector of length M containing cumulative percentage of eigenvalues that are greater than or equal to CON.

Method:

TRACE divides each eigenvalue greater than or equal to CON by M and adds the result to the previous total to obtain the cumulative percentage for each eigenvalue.

2.9.3 VARMX Subroutine

VARMX performs orthogonal rotations of a factor matrix. This subroutine normally occurs in a sequence of calls to subroutines CORRE, EIGEN, TRACE, LOAD, and VARMX in the performance of a factor analysis (Test: FACTO.ext).

Usage: CALL VARMX(M,K,A,NC,TV,H,F,D,IER)

Parameter	Description
M	Number of variables and number of rows of matrix A.
K	Number of factors.
A	Input is the original factor matrix, and output is the rotated factor matrix. The order of matrix A is M by K.
NC	Output variable containing the number of iteration cycles performed.
TV	Output vector containing the variance of the factor matrix for each iteration cycle. The variance prior to the first iteration cycle is also calculated. This means that NC+1 variances are stored in vector TV. Maximum number of iteration cycles allowed in this subroutine is 50; therefore, the length of vector TV is 51.
H	Output vector of length M containing the original communalities.
F	Output vector of length M containing the final communalities.
D	Output vector of length M containing the differences between the original and final communalities.
IER	Resultant error code: 0 No error. 1 Convergence was not achieved in 50 cycles of rotation.

Remarks:

If the variance computed after each iteration cycle does not increase for four successive times, VARMX stops rotation.

Method:

Kaiser's varimax rotation as described in COMPUTER PROGRAM FOR VARIMAX ROTATION IN FACTOR ANALYSIS, Educational and Psychological Measurement, Vol. XIX, No. 3, 1959.

2.10 TIME SERIES SUBROUTINES

The sections that follow describe the time series subroutines.

2.10.1 AUTO Subroutine

AUTO finds the autocovariances of a series A for lags 0 to L-1 (Test: TIMSER.ext).

Usage: CALL AUTO(A,N,L,R)

Parameter	Description
A	Input vector of length N containing the time series whose autocovariance is desired.
N	Length of the vector A.
L	Autocovariance is calculated for lags of 0, 1, 2,..., L-1.
R	Output vector of length L containing autocovariances of series A.

Remarks:

The length of R must be different from the length of A. N must be greater than L. If not, R(1) is set to zero and return is made to the calling program.

Method:

Refer to Blackman and Tukey 1959.

2.10.2 CROSS Subroutine

CROSS finds the cross-covariances of a series A with a series B that leads and lags A (Test: TIMSER.ext).

Usage: CALL CROSS(A,B,N,L,R,S)

Parameter	Description
A	Input vector of length N containing first time series.
B	Input vector of length N containing second time series.
N	Length of series A and B.
L	Cross-covariance is calculated for lags and leads of 0, 1, 2, ..., L-1.
R	Output vector of length L containing cross-covariances of A with B, where B lags A.
S	Output vector of length L containing cross-covariances of A with B, where B leads A.

Remarks:

N must be greater than L; if it is not, CROSS sets R(1) and S(1) to zero and returns to the calling program.

Method:

Refer to Blackman and Tukey 1959.

2.10.3 EXSMO Subroutine

EXSMO finds the triple exponential smoothed series S of the given series X (Test: EXPON.ext).

Usage: CALL EXSMO(X,NX,AL,A,B,C,S)

Parameter	Description
X	Input vector of length NX containing time series data which is to be exponentially smoothed.
NX	The number of elements in X.
AL	Smoothing constant, alpha. AL must be greater than 0 and less than 1.
A,B,C	Coefficients of the prediction equation, where S is predicted T periods hence by A+B*T+C*T*T/2. As input--if A=B=C=0, the program will provide initial values. If at least one of A, B, C is not zero, the program will take given as initial values. As output--A, B, C contain the latest, updated coefficients of prediction.
S	Output vector of length NX containing triple exponentially smoothed time series.

Method:

Refer to Brown 1963.

2.10.4 SMO Subroutine

SMO smooths or filters series A by weights W (Test: TIMSER.ext).

Usage: CALL SMO(A,N,W,M,L,R)

Parameter	Description
A	Input vector of length N containing time series data.
N	Length of series A.
W	Input vector of length M containing weights.
M	Number of items in weight vector. M must be an odd integer. If M is an even integer, any fraction resulting from the calculation of (L*(M-1))/2 in (1) and (2) below will be truncated.
L	Selection integer. For example, L=12 means that weights are applied to every 12th item of A. L=1 applies weights to successive items of A. For monthly data, L=12 gives year-to-year averages and L=1 gives month-to-month averages.
R	Output vector of length N. From IL to IH elements of the vector R are filled with the smoothed series and other elements with zero, where: IL=(L*(M-1))/2+1...(1) IH=N-(L*(M-1))/2...(2)

Remarks:

N must be greater than or equal to the product L*M.

Method:

Refer to Healy and Bogert 1963.

2.11 NONPARAMETRIC STATISTICAL SUBROUTINES

The sections that follow describe the nonparametric statistical subroutines.

2.11.1 CHISQ Subroutine

CHISQ computes the chi-square from a contingency table (Test: NONPAR.ext).

Usage: CALL CHISQ(A,N,M,CS,NDF,IER,TR,TC)

Parameter	Description
A	Input matrix, N by M, containing contingency table.
N	Number of rows in A.
M	Number of columns in A.
CS	Chi-square (output).
NDF	Number of degrees of freedom (output).
IER	Resultant error code: 0 No error. 1 Expected value is less than 1.0 in one or more cells. 3 Number of degrees of freedom is zero.
TR	Work vector of length N.
TC	Work vector of length M.

Remarks:

If one or more cells contain an expected value (that is, theoretical value) less than 1.0, CHISQ computes chi-square, and sets the error code to 1. (See reference given below.) CHISQ sets chi-square to zero if either N or M is 1 (error code 3).

Method:

Refer to Siegel 1956.

2.11.2 KRANK Subroutine

KRANK tests the correlation between two variables by the Kendall rank correlation coefficient (Test: NONPAR.ext).

Usage: CALL KRANK(A,B,R,N,TAU,SD,Z,NR)

Parameter	Description
A	Input vector of N observations for first variable.
B	Input vector of N observations for second variable.
R	Output vector of ranked data of length 2*N. Smallest observation is ranked 1, largest is ranked N. Ties are assigned average of tied ranks.
N	Number of observations.
TAU	Kendall rank correlation coefficient (output).
SD	Standard deviation (output).
Z	Test of significance of TAU in terms of normal distribution (output).
NR	Code, 0 for unranked data in A and B, 1 for ranked data in A and B (input).

Remarks:

SD and Z are set to zero if N is less than 10.

Subroutines and Function Subprograms Required:

RANK
TIE

Method:

Refer to Siegel 1956.

2.11.3 QTEST Subroutine

QTEST tests whether three or more matched groups of dichotomous data differ significantly by the Cochran Q-test (Test: NPAR2.ext).

Usage: CALL QTEST(A,N,M,Q,NDF)

Parameter	Description
A	Input matrix, N by M, of dichotomous data (0 and 1).
N	Number of sets in each group.
M	Number of groups.
Q	Cochran Q statistic (output).
NDF	Number of degrees of freedom (output).

Remarks:

M must be 3 or greater.

Method:

Refer to Siegel 1956.

2.11.4 RANK Subroutine

RANK ranks a vector of values. RANK assigns tied values to the average rank (Tests: NONPAR.ext, NPAR2.ext).

Usage: CALL RANK(A,R,N)

Parameter	Description
A	Input vector of N values.
R	Output vector of length N. Smallest value is ranked 1, largest is ranked N. Ties are assigned average of tied ranks.
N	Number of values.

Method:

RANK searches vector A for successively larger elements. RANK locates all ties and computes their value. For example, if two values are tied for sixth rank, RANK assigns each of them a rank of 6.5 ((6+7)/2).

2.11.5 SRANK Subroutine

SRANK tests the correlation between two variables by the Spearman rank correlation coefficient (Test: NONPAR.ext).

Usage: CALL SRANK(A,B,R,N,RS,T,NDF,NR)

Parameter	Description
A	Input vector of N observations for first variable.
B	Input vector of N observations for second variable.
R	Output vector for ranked data, length is 2*N. Smallest observation is ranked 1, largest is ranked N. Ties are assigned average of tied ranks.
N	Number of observations.
RS	Spearman rank correlation coefficient (output).
T	Test of significance of RS (output).
NDF	Number of degrees of freedom (output).
NR	Code, 0 for unranked data in A and B, 1 for ranked data in A and B (input).

Remarks:

SRANK sets T to 0 if N is less than 10.

Subroutines and Function Subprograms Required:

RANK
TIE

Method:

Refer to Siegel 1956.

2.11.6 TIE Subroutine

TIE calculates the correction factor due to ties (Tests: NONPAR.ext, NPAR2.ext).

Usage: CALL TIE(R,N,KT,T)

Parameter	Description
R	Input vector of ranks of length N containing values 1 to N.
N	Number of ranked values.
KT	Input code for calculation of correction factor: 1 solve equation 1 2 solve equation 2
T	Correction factor (output): Equation 1 T=SUM(CT**3-CT)/12 Equation 2 T=SUM(CT*(CT-1)/2) where CT is the number of observations tied for a given rank.

Method:

TIE searches vector R for successively larger ranks. TIE counts ties and some correction factor 1 of 2.

2.11.7 TWOAV Subroutine

TWOAV tests whether samples are from the same population by the Friedman two-way analysis of variance test (Test: NONPAR.ext).

Usage: CALL TWOAV(A,R,N,M,W,XR,NDF,NR)

Parameter	Description
A	Input matrix, N by M, of original data.
R	Output matrix, N by M, of ranked data.
N	Number of groups.

Parameter	Description
M	Number of cases in each group.
W	Work area of length 2*M.
XR	Friedman statistic (output).
NDF	Number of degrees of freedom (output).
NR	Code, 0 for unranked data in A, 1 for ranked data in A (input).

Subroutines and Function Subprograms Required:

RANK

Method:

Refer to Siegel 1956.

2.11.8 UTEST Subroutine

UTEST determines whether two independent groups are from the same population by the Mann-Whitney U-test (Test: NONPAR.ext).

Usage: CALL UTEST(A,R,N1,N2,U,Z,IER)

Parameter	Description
A	Input vector of cases consisting of two independent groups. Smaller group precedes larger group. Length is N1+N2.
R	Output vector of ranks. Smallest value is ranked 1, largest is ranked N. Ties are assigned average of tied ranks. Length is N1+N2.
N1	Number of cases in smaller group.
N2	Number of cases in larger group.
U	Statistic used to test homogeneity of the two groups (output).
Z	Measure of significance of U in terms of normal distribution (output).
IER	Resultant error code: 0 No error. 1 All values of one group are tied.

Remarks:

Z is set to zero if N2 is less than 20.

Subroutines and Function Subprograms Required:

RANK
TIE

Method:

Refer to Siegel 1956.

2.11.9 WTEST Subroutine

WTEST uses the Kendall coefficient of concordance to test the degree of association among a number of variables (Test: NPAR2.ext).

Usage: CALL WTEST(A,R,N,M,WA,W,CS,NDF,NR)

Parameter	Description
A	Input matrix, N by M, of original data.
R	Output matrix, N by M, of ranked data. Smallest value is ranked 1, largest is ranked N. Ties are assigned average of tied ranks.
N	Number of variables.
M	Number of cases.
WA	Work area vector of length 2*M.
W	Kendall coefficient of concordance (output).
CS	Chi-square (output).
NDF	Number of degrees of freedom (output).
NR	Code, 0 for unranked data in A, 1 for ranked data in A (input).

Remarks:

WTEST sets chi-square to 0 if M is 7 or smaller.

Subroutines and Function Subprograms Required:

RANK
TIE

Method:

Refer to Siegel 1956.

2.12 RANDOM NUMBER GENERATOR SUBROUTINE

The section that follows describes the random number generator subroutine.

2.12.1 GAUSS Subroutine

GAUSS computes a normally distributed random number with a given mean and standard deviation (Test: NONPAR.ext).

NOTE

The uniform random number generator RANDU is not provided with this package, because the FORTRAN IV and FORTRAN 77 libraries contain a generator of the same type.

Usage: CALL GAUSS(IX,S,AM,V)

Parameter	Description
IX	IX is an integer dimensioned to two. On the first entry to GAUSS, both elements of the IX array should be equal to zero. Thereafter, they will contain a uniformly distributed integer random number generated by the subroutine for use on the next entry to the subroutine.
S	The desired standard deviation of the normal distribution.
AM	The desired mean of the normal distribution.
V	The value of the computed normal random variable.

Remarks:

Subroutines and Function Subprograms Required:

RANDU

Method:

GAUSS uses 12 uniform random numbers to compute normal random numbers by the central limit theorem. GAUSS then adjusts the result to match the given mean and standard deviation. GAUSS finds the uniform random numbers computed within the subroutine by the power residue method.

2.13 REFERENCES

Anderson, T.W. Introduction to Multivariate Statistical Analysis. New York: John Wiley and Sons, 1958.

Blackman, R.B., and J.W. Tukey. The Measurement of Power Spectra. New York: Dover Publications, 1959.

Brown, R.G. Smoothing, Forecasting and Prediction of Discrete Time Series. New Jersey: Prentice-Hall, 1963.

Cooley, W.W., and P.R. Lohnes. Multivariate Procedures for the Behavioral Sciences. New York: John Wiley and Sons, 1962.

Dixon, W.J., ed. BMDP Computer Programs Manual. Los Angeles: UCLA, 1977.

Healey, J.R., and B.P. Bogert. "FORTRAN Subroutines for Time Series Analysis," Communications of ACM, Vol. 6. No. 1 (January, 1963).

Kaiser. "Computer Program for Varimax Rotation in Factor Analysis" Educational and Psychological Measurement, Vol. 11, No. 3 (1959).

Kendall, M.G. The Advance Theory of Statistics, Vol. 1. New York: Hafner Press, 1977.

Ostle, B. Statistics in Research. Ames: Iowa State College Press, 1975.

Ralston, A., and H. Wilf, eds. Mathematical Methods for Digital Computers. New York: John Wiley and Sons, 1962.

Siegal, S. Nonparametric Statistics for the Behavioral Sciences. New York: McGraw-Hill, 1956.

CHAPTER 3

MATHEMATICAL OPERATIONS

3.1 SPECIAL MATRIX OPERATIONS SUBROUTINES

The sections that follow describe the special matrix operations subroutines.

3.1.1 EIGEN Subroutine

EIGEN computes the eigenvalues and eigenvectors of a real, symmetric matrix (Test: FACTO.ext, MCANO.ext).

Usage: CALL EIGEN(A,R,N,MV)

Parameter	Description
A	Original matrix (symmetric), destroyed in computation. Resultant eigenvalues are developed in diagonal of matrix A in descending order.
R	Resultant matrix of eigenvectors (stored column-wise, in same sequence as eigenvalues).
N	Order of matrices A and R.
MV	Input code: 0 Compute eigenvalues and eigenvectors. 1 Compute eigenvalues only (R need not be dimensioned but must still appear in calling sequence).

Remarks:

The original matrix A must be real symmetric (storage mode of 1). Matrix A cannot be in the same location as matrix B.

Method:

EIGEN uses the diagonalization method originated by Jacobi and adapted by Von Neumann for large computers (Ralston and Wilf, eds. 1960).

3.1.2 MINV Subroutine

MINV inverts a matrix and calculates its determinant. A determinant of zero indicates that the matrix is singular (Tests: MCANO.ext, MDISC.ext, POLRG.ext, SOLVEN.ext).

Usage: CALL MINV(A,N,D,L,M)

Parameter	Description
A	Input matrix, destroyed in computation and replaced by resultant inverse.
N	Order of matrix A.
D	Resultant determinant.
L	Work vector of length N.
M	Work vector of length N.

Remarks:

Matrix A must be a general matrix.

Method:

MINV uses the standard Gauss-Jordan method. For accuracy, MINV employs both row and column pivoting.

3.2 MATRIX SUBROUTINES

The sections that follow describe the matrix subroutines.

3.2.1 ARRAY Subroutine

ARRAY converts a data array from single to double dimension or vice versa. ARRAY links the user program that has double dimension arrays and subroutines that operate on arrays of data in a vector fashion (Test: MACHK1.ext).

Usage: CALL ARRAY(MODE,I,J,N,M,S,D)

Parameter	Description
MODE	Code indicating type of conversion: 1 From single to double dimension. 2 From double to single dimension.
I	Number of rows in actual data matrix.
J	Number of columns in actual data matrix.
N	Number of rows specified for the matrix D in dimension statement.
M	Number of columns specified for the matrix D in dimension statement.

Parameter	Description
S	If MODE=1, this vector containing the elements of a data matrix of size I by J is input. Column I+1 of data matrix follows column I, and so forth. If MODE=2, this vector is output representing a data matrix of size I by J containing its columns consecutively. The length of S is IJ, where IJ=I*J.
D	If MODE=1, this matrix of size N by M is output containing a data matrix of size I by J in the first I rows and J columns. If MODE=2, this N by M matrix is input containing a data matrix of size I by J in the first I rows and J columns.

Remarks:

Vector S can be in the same location as matrix D. Vector S is referred to as a matrix in other scientific subroutines because it contains a data matrix. This subroutine only converts general data matrices (storage mode of 0).

Method:

Refer to the discussion on variable data size in the Section 1.1.1.

3.2.2 CADD Subroutine

CADD adds a column of one matrix to a column of another matrix (Test: MACHK4.ext).

Usage: CALL CADD(A,ICA,R,ICR,N,M,MS,L)

Parameter	Description
A	Name of input matrix.
ICA	Column in matrix A to be added to column ICR of R.
R	Name of output matrix.
ICR	Column in matrix R where summation is developed.
N	Number of rows in A and R.
M	Number of columns in A.
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.
L	Number of columns in R.

Remarks:

Matrix R must be a general matrix and cannot be in the same location as matrix A, unless A is general.

Subroutines and Function Subprograms Required:

LOC

Method:

CADD adds each element of column ICA of matrix A to the corresponding element of column ICR of matrix R.

3.2.3 CCPY Subroutine

CCPY copies a column of a matrix into a vector (Tests: MACHK3.ext, MACHK4.ext).

Usage: CALL CCPY(A,L,R,N,M,MS)

Parameter	Description
A	Name of input matrix.
L	Column of A to be moved to R.
R	Name of output vector of length N.
N	Number of rows in A.
M	Number of columns in A.
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.

Subroutines and Function Subprograms Required:

LOC

Method:

CCPY moves elements of column L to corresponding positions of vector R.

3.2.4 CCUT Subroutine

CCUT partitions a matrix between specified columns to form two resultant matrices (Test: MACHK2.ext).

Usage: CALL CCUT(A,L,R,S,N,M,MS)

Parameter	Description
A	Name of input matrix.
L	Column of A to the left of which partitioning takes place.
R	Name of matrix to be formed from left portion of A.

Parameter	Description
S	Name of matrix to be formed from right portion of A.
N	Number of rows in A.
M	Number of columns in A.
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

Matrix R and matrix S must be general matrices. Neither matrix R nor matrix S can be in the same location as matrix A. Matrix R cannot be in the same location as matrix S.

Subroutines and Function Subprograms Required:

LOC

Method:

CCUT moves elements of matrix A to the left of column L to form matrix R of N rows and L-1 columns. CCUT moves elements of matrix A in column L and to the right of L to form matrix S of N rows and M-L+1 columns.

3.2.5 CINT Subroutine

CINT interchanges two columns of a matrix (Test: COLROW.ext).

Usage: CALL CINT(A,N,LA,LB)

Parameter	Description
A	Name of matrix.
N	Number of rows in A.
LA	Column to be interchanged with column LB.
LB	Column to be interchanged with column LA.

Remarks:

Matrix A must be a general matrix.

Method:

CINT interchanges each element of column LA with the corresponding element of column LB.

3.2.6 CSRT Subroutine

CSRT sorts columns of a matrix (Test: MACHK4.ext).

Usage: CALL CSRT(A,B,R,N,M,MS)

Parameter	Description
A	Name of input matrix to be sorted.
B	Name of input vector which contains sorting key.
R	Name of sorted output matrix.
N	Number of rows in A and R.
M	Number of columns in A and R and length of B.
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

Matrix R must be a general matrix and cannot be in the same location as matrix A. M must be greater than 1. N must be greater than or equal to 2.

Subroutines and Function Subprograms Required:

LOC
CCPY

Method:

CSRT sorts the columns of input matrix A to form output matrix R. CSRT determines the sorted column sequence by the values of elements in row vector B.

The lowest valued element in B causes the corresponding column of A to be placed in the first column of R. The highest valued element of B causes the corresponding row of A to be placed in the last column of R. If duplicate values exist in B, CSRT moves the corresponding columns of A to R in the same order as in A.

3.2.7 CSUM Subroutine

CSUM sums elements of each column to form a row vector (Test: MACHK3.ext).

Usage: CALL CSUM(A,R,N,M,MS)

Parameter	Description
A	Name of input matrix.
R	Name of vector of length M.
N	Number of rows in A.

Parameter	Description
M	Number of columns in A.
MS	One-digit number for storage mode of matrix A. 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

Vector R cannot be in the same location as matrix A unless A is general.

Subroutines and Function Subprograms Required:

LOC

Method:

CSUM sums elements down each column into a corresponding element of output row vector R.

3.2.8 CTAB Subroutine

CTAB tabulates the columns of data based on the key contained in vector B. It adds columns of the original matrix into a new matrix in the columns specified by the floating point number in the respective row of the vector B (Test: MACHK4.ext).

Usage: CALL CTAB(A,B,R,S,N,M,MS,L)

Parameter	Description
A	Name of input matrix.
B	Name of input vector of length M containing key.
R	Name of output matrix containing summary of column data. It is initially set to zero by this subroutine.
S	Name of output vector of length L+1 containing counts.
N	Number of rows in A and R.
M	Number of columns in A.
L	Number of columns in R.
MS	One-digit number of storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

Matrix R must be a general matrix.

Subroutines and Function Subprograms Required:

LOC
CADD

Method:

CTAB tabulates the columns of data in matrix A based on the key contained in vector B. The floating point number in B(I) is truncated to form J. The Ith column of A is added to the Jth column of matrix R and 1 is added to S(J). If the value of J is not between 1 and L, 1 is added to S(L+1).

Upon completion, the output matrix R contains a summary of column data as specified by vector B. Each element in vector S contains a count of the number of columns of A used to form R. Element S(L+1) contains the number of columns of A not included in R as a result of J being less than 1 or greater than L.

3.2.9 CTIE Subroutine

CTIE adjoins two matrices with the same row dimensions to form one resultant matrix (Test: MACHK2.ext).

Usage: CALL CTIE(A,B,R,N,M,MSA,MSB,L)

Parameter	Description
A	Name of first input matrix.
B	Name of second input matrix.
R	Name of output matrix.
N	Number of rows in A, B, R.
M	Number of columns in A.
MSA	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.
MSB	Same as MSA except for matrix B.
L	Number of columns in B.

Remarks:

Matrix R must be a general matrix and cannot be in the same location as either matrix A or matrix B. Matrix A must have the same number of rows as matrix B.

Subroutines and Function Subprograms Required:

LOC

Method:

Matrix B is attached to the right of matrix A. The resultant matrix R contains N rows and M+L columns.

3.2.10 DCLA Subroutine

DCLA sets each diagonal element of a matrix equal to a scalar (Test: MACHK2.ext).

Usage: CALL DCLA(A,C,N,MS)

Parameter	Description
A	Name of input matrix.
C	Scalar. (Single precision floating point variables.)
N	Number of rows and columns in matrix A.
MS	One-digit number for storage mode of matrix A. 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

Input matrix must be a square matrix.

Subroutines and Function Subprograms Required:

LOC

Method:

DCLA replaces each element on the diagonal of a matrix with a scalar C.

3.2.11 DCPY Subroutine

DCPY copies diagonal elements of a matrix into a vector (Test: MACHK2.ext).

Usage: CALL DCPY(A,R,N,MS)

Parameter	Description
A	Name of input matrix.
R	Name of output vector of length N.
N	Number of rows and columns in matrix A.
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

The input matrix must be a square matrix.

Subroutines and Function Subprograms Required:

LOC

Method:

DCPY moves the elements on diagonal of matrix to corresponding positions of vector R.

3.2.12 GMADD Subroutine

GMADD adds two general matrices (mode 0) to form resultant matrix (Test: MACHK1.ext).

Usage: CALL GMADD(A,B,R,N,M)

Parameter	Description
A	Name of first input matrix.
B	Name of second input matrix.
R	Name of output matrix.
N	Number of rows in A, B, R.
M	Number of columns in A, B, R.

Remarks:

All matrices must be stored as general matrices.

Method:

GMADD performs addition element by element.

3.2.13 GMPRD Subroutine

GMPRD multiplies two general matrices to form a resultant general matrix (Tests: MACHK1.ext, SOLVEN.ext).

Usage: CALL GMPRD(A,B,R,N,M,L)

Parameter	Description
A	Name of first input matrix.
B	Name of second input matrix.
R	Name of output matrix.
N	Number of rows in A.
M	Number of columns in A and rows in B.
L	Number of columns in B.

Remarks:

All matrices must be stored as general matrices. Matrix R cannot be in the same location as either matrix A or matrix B. The number of columns of matrix A must be equal to the number of rows in matrix B.

Method:

GMPRD premultiplies the M by L matrix B by the N by M matrix A and stores the result in the N by L matrix R.

3.2.14 GMSUB Subroutine

GMSUB subtracts one general matrix from another to form a resultant matrix (Test: MACHK1.ext).

Usage: CALL GMSUB(A,B,R,N,M)

Parameter	Description
A	Name of first input matrix.
B	Name of second input matrix.
R	Name of output matrix.
N	Number of rows in A, B, R.
M	Number of columns in A, B, R.

Remarks:

All matrices must be stored as general matrices.

Method:

GMSUB subtracts matrix B elements from corresponding matrix A elements.

3.2.15 GMTRA Subroutine

GMTRA transposes a general matrix (Test: MACHK1.ext).

Usage: CALL GMTRA(A,R,N,M)

Parameter	Description
A	Name of matrix to be transposed.
R	Name of resultant matrix.
N	Number of rows in A and columns in R.
M	Number of columns in A and rows in R.

Remarks:

Matrix A and matrix R must be stored as general matrices.

Method:

GMTRA transposes N by M matrix A to form M by N matrix R.

3.2.16 GTPRD Subroutine

GTPRD premultiplies a general matrix by the transpose of another general matrix (Test: MACHK1.ext).

Usage: CALL GTPRD (A,B,R,N,M,L)

Parameter	Description
A	Name of first input matrix.
B	Name of second input matrix.
R	Name of output matrix.
N	Number of rows in A and B.
M	Number of columns in A and rows in R.
L	Number of columns in B and R.

Remarks:

All matrices must be stored as general matrices. Matrix R cannot be in the same location as either matrix A or matrix B.

Method:

GTPRD does not actually calculate the matrix transpose of A. Instead, it takes the elements of matrix A column wise (rather than row wise) for postmultiplication by matrix B.

3.2.17 LOC Subroutine

LOC computes a vector subscript for an element in a matrix of specified storage mode (Tests: DASCR.ext, MACHK2.ext).

Usage: CALL LOC(I,J,IR,N,M,MS)

Parameter	Description
I	Row number of element.
J	Column number of element.
IR	Resultant vector subscript.
N	Number of rows in matrix.
M	Number of columns in matrix.
MS	One-digit number for storage mode of matrix: 0 General. 1 Symmetric. 2 Diagonal.

Method:

MS=0	Subscript is computed for a matrix with N*M elements in storage (general matrix).
MS=1	Subscript is computed for a matrix with N*(N+1)/2 in storage (upper triangle of symmetric matrix). If element is in lower triangular portion, subscript is corresponding element in upper triangle.
MS=2	Subscript is computed for a matrix with N elements in storage (diagonal elements of diagonal matrix). If element is not on diagonal (and therefore not in storage), IR is set to zero.

3.2.18 MADD Subroutine

MADD adds two matrices (of any mode) to form resultant matrix (Test: MACHK1.ext).

Usage: CALL MADD(A,B,R,N,M,MSA,MSB)

Parameter	Description
A	Name of first input matrix.
B	Name of second input matrix.
R	Name of output matrix.
N	Number of rows in A, B, R.
M	Number of columns in A, B, R.
MSA	One-digit number of storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.
MSB	One-digit number of storage mode of matrix B: 0 General. 1 Symmetric. 2 Diagonal.

Subroutines and Function Subprograms Required:

LOC

Method:

MADD first determines storage mode of output matrix, then adds the corresponding elements.

The storage mode of the output matrix for all combinations of input matrices is as follows:

A	B	R
general	general	general
general	symmetric	general
general	diagonal	general
symmetric	general	general
symmetric	symmetric	symmetric
symmetric	diagonal	symmetric
diagonal	general	general
diagonal	symmetric	symmetric
diagonal	diagonal	diagonal

3.2.19 MATA Subroutine

MATA premultiplies a matrix by its transpose to form a symmetric matrix (Test: MACHK2.ext).

Usage: CALL MATA(A,R,N,M,MS)

Parameter	Description
A	Name of input matrix.
R	Name of output matrix.
N	Number of rows in A.
M	Number of columns in A. Also number of rows and number of columns in R.
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

Matrix R must be a symmetric matrix with a storage mode of 1. Matrix R cannot be in the same location as matrix A.

Subroutines and Function Subprograms Required:

LOC

Method:

Calculation of (A transpose A) results in a symmetric matrix regardless of the storage mode of the input matrix. The elements of matrix A are not changed.

3.2.20 MCPY Subroutine

MCPY copies an entire matrix (Tests: COLROW.ext, MACHK2.ext).

Usage: CALL MCPY(A,R,N,M,MS)

Parameter	Description
A	Name of input matrix.
R	Name of output matrix.
N	Number of rows in A or R.
M	Number of columns in A or R.
MS	One-digit number for storage mode of matrix A and R: 0 General. 1 Symmetric. 2 Diagonal.

Subroutines and Function Subprograms Required:

LOC

Method:

MCPY moves each element of matrix A to the corresponding element of matrix R.

3.2.21 MFUN Subroutine

MFUN applies a function to each element of a matrix to form a resultant matrix (Tests: MACHK3.ext, MACHK4.ext).

Usage: CALL MFUN(A,F,R,N,M,MS)

Parameter	Description
A	Name of input matrix.
F	Name of FORTRAN-furnished or user function subprogram.
R	Name of output matrix.
N	Number of rows in matrices A and R.
M	Number of columns in matrices A and R.
MS	One-digit number for storage mode of matrix A and R: 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

An external statement must precede the CALL statement to identify parameter F as the name of a function. Precision depends on precision of function used.

Subroutines and Function Subprograms Required:

LOC

Method:

MFUN applies function F to each element of matrix A to form matrix R.

3.2.22 MPRD Subroutine

MPRD multiplies two matrices to form a resultant matrix (Test: MACHK4.ext).

Usage: CALL MPRD(A,B,R,N,M,MSA,MSB,L)

Parameter	Description
A	Name of first input matrix.
B	Name of second input matrix.
R	Name of output matrix.
N	Number of rows in A and R.
M	Number of columns in A and rows in B.
MSA	One-digit number of storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.
MSB	One-digit number of storage mode of matrix B: 0 General. 1 Symmetric. 2 Diagonal.
L	Number of columns in B and R.

Remarks:

Matrix R cannot be in the same location as either matrix A or matrix B. The number of columns of matrix A must be equal to number of rows of matrix B.

Subroutines and Function Subprograms Required:

LOC

Method:

MPRD premultiplies the M by L matrix B by the N by M matrix A and stores the result in the N by L matrix R. This is a row into column product.

The storage mode of the output matrix for all combinations of input matrices is as follows:

A	B	R
general	general	general
general	symmetric	general
general	diagonal	general
symmetric	general	general
symmetric	symmetric	general
symmetric	diagonal	general
diagonal	general	general
diagonal	symmetric	general
diagonal	diagonal	diagonal

3.2.23 MSTR Subroutine

MSTR changes storage mode of a matrix (Test: MACHK4.ext).

Usage: CALL MSTR(A,R,N,MSA,MSR)

Parameter	Description
A	Name of input matrix.
R	Name of output matrix.
N	Number of rows and columns in A and R.
MSA	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.
MSR	One-digit number for storage mode of matrix R: 0 General. 1 Symmetric 2 Diagonal.

Remarks:

Matrix A must be a square matrix. Matrix R cannot be in the same location as matrix A.

Subroutines and Function Subprograms Required:

LOC

Method:

MSTR restructures matrix A to form matrix R.

MSA	MSR	
0	0	Matrix A is moved to matrix R.
0	1	The upper triangle elements of a general matrix are used to form a symmetric matrix.

MSA	MSR	
0	2	The diagonal elements of a general matrix are used to form a diagonal matrix.
1	0	A symmetric matrix is expanded to form a general matrix.
1	1	Matrix A is moved to matrix R.
1	2	The diagonal elements of a symmetric matrix are used to form a diagonal matrix.
2	0	A diagonal matrix is expanded by inserting missing zero elements to form a general matrix.
2	1	A diagonal matrix is expanded by inserting missing zero elements to form a symmetric matrix.
2	2	Matrix A is moved to matrix R.

3.2.24 MSUB Subroutine

MSUB subtracts one matrix from another, element by element, to form a resultant matrix (Test: MACHK3.ext).

Usage: CALL MSUB(A,B,R,N,M,MSA,MSB)

Parameter	Description
A	Name of first input matrix.
B	Name of second input matrix.
R	Name of output matrix.
N	Number of rows in A, B, R.
M	Number of columns in A, B, R.
MSA	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.
MSB	One-digit number for storage mode of matrix B: 0 General. 1 Symmetric. 2 Diagonal.

Subroutines and Function Subprograms Required:

LOC

Method:

MSUB first determines the structure of the output matrix, then it subtracts matrix B elements from corresponding matrix A elements.

The storage mode of the output matrix for all combinations of input matrices is as follows:

A	B	R
general	general	general
general	symmetric	general
general	diagonal	general
symmetric	general	general
symmetric	symmetric	symmetric
symmetric	diagonal	symmetric
diagonal	general	general
diagonal	symmetric	symmetric
diagonal	diagonal	diagonal

3.2.25 MTRA Subroutine

MTRA transposes a matrix (of any mode) (Test: COLROW.ext).

Usage: CALL MTRA(A,R,N,M,MS)

Parameter	Description
A	Name of matrix to be transposed.
R	Name of output matrix.
N	Number of rows in A and columns in R.
M	Number of columns in A and rows in R.
MS	One-digit number for storage mode of matrix A and R: 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

Matrix R cannot be in the same location as matrix A.

Subroutines and function Subprograms Required:

MCPY

Method:

MTRA transposes N by M matrix A to form M by N matrix R by moving each row of A into the corresponding column of R. If matrix A is symmetric or diagonal, then matrix R is the same as A.

3.2.26 RADD Subroutine

RADD adds row of one matrix to row of another matrix (Test: MACHK4.ext).

Usage: CALL RADD(A,IRA,R,IRR,N,M,MS,L)

Parameter	Description
A	Name of input matrix.
IRA	Row in matrix A to be added to row IRR of matrix R.
R	Name of output matrix.
IRR	Row in matrix R where summation is developed.
N	Number of rows in A.
M	Number of columns in A and R.
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.
L	Number of rows in R.

Remarks:

Matrix R must be a general matrix. Matrix R cannot be in the same location as matrix A, unless A is general.

Subroutines and Function Subprograms Required:

LOC

Method:

RADD adds each element of row IRA of matrix A to corresponding element of row IRR of matrix R.

3.2.27 RCPY Subroutine

RCPY copies a row of a matrix into a vector (Test: MACHK4.ext).

Usage: CALL RCPY(A,L,R,N,M,MS)

Parameter	Description
A	Name of input matrix.
L	Row of A to be moved to R.
R	Name of output vector of length M.
N	Number of rows in A.
M	Number of columns in A.
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.

Subroutines and Function Subprograms Required:

LOC

Method:

RCPY moves elements of row L to corresponding positions of vector R.

3.2.28 RCUT Subroutine

RCUT partitions a matrix between specified rows to form two resultant matrices (Test: MACHK2.ext).

Usage: CALL RCUT(A,L,R,S,N,M,MS)

Parameter	Description
A	Name of input matrix.
L	Row of A above which partitioning takes place.
R	Name of matrix to be formed from upper portion of A.
S	Name of matrix to be formed from lower portion of A.
N	Number of rows in A.
M	Number of columns in A.
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

Matrix R and matrix S must be general matrices. Neither matrix R nor matrix S can be in the same location as matrix A. Matrix R cannot be in the same location as matrix S.

Subroutines and Function Subprograms Required:

LOC

Method:

RCUT moves elements of matrix A above row L to form matrix R of L-1 rows and M columns. RCUT moves elements of matrix A in row L and below to form matrix S of N-L+1 rows and M columns.

3.2.29 RECP Subroutine

RECP calculates the reciprocal of an element. This is a FORTRAN function subprogram that can be used as an argument by subroutine MFUN (Test: MACHK4.ext).

Usage: CALL RECP(E)

Parameter	Description
E	Matrix element.

Remarks:

Reciprocal of zero is taken to be 1.0E75.

Method:

Reciprocal of element E is placed in RECP.

3.2.30 RINT Subroutine

RINT interchanges two rows of a matrix (Test: COLROW.ext).

Usage: CALL RINT(A,N,M,LA,LB)

Parameter	Description
A	Name of matrix.
N	Number of rows in A.
M	Number of columns in A.
LA	Row to be interchanged with row LB.
LB	Row to be interchanged with row LA.

Remarks:

Matrix A must be a general matrix.

Method:

RINT interchanges each element of row LA with the corresponding element of row LB.

3.2.31 RSRT Subroutine

RSRT sorts rows of a matrix (Test: MACHK3.ext).

Usage: CALL RSRT(A,B,R,N,M,MS)

Parameter	Description
A	Name of input matrix to be sorted.
B	Name of input vector which contains sorting key.
R	Name of sorted output matrix.
N	Number of rows in A and R and length of B.
M	Number of columns in A and R.

Parameter	Description
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

Matrix R must be a general matrix and cannot be in the same location as matrix A. N must be greater than 1. M must be greater than or equal to 2.

Subroutines and Function Subprograms Required:

LOC

Method:

RSRT sorts rows of input matrix A to form output matrix R. RSRT determines the sorted row sequence by the values of the elements in column vector B. The lowest valued element in B causes the corresponding row of A to be placed in the first row of R. The highest valued element of B causes the corresponding row of A to be placed in the last row of R. If duplicate values exist in B, RSRT moves the corresponding rows of A to R in the same order as in A.

3.2.32 RSUM Subroutine

RSUM sums elements of each row to form a column vector (Test: MACHK3.ext).

Usage: CALL RSUM(A,R,N,M,MS)

Parameter	Description
A	Name of input matrix.
R	Name of vector of length N.
N	Number of rows in A.
M	Number of columns in A.
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

Vector R cannot be in the same location as matrix A unless A is general.

Subroutines and Function Subprograms Required:

LOC

Method:

RSUM sums elements across each row into a corresponding element of output column vector R.

3.2.33 RTAB Subroutine

RTAB tabulates the rows of data based on the key contained in the vector B. Rows of the original matrix are added into a new matrix in the rows specified by the floating-point number in the respective row of the vector B (Test: MACHK4.ext).

Usage: CALL RTAB(A,B,R,S,N,M,L,MS)

Parameter	Description
A	Name of input matrix.
B	Name of input vector of length N containing key.
R	Name of output matrix containing summary of row data. It is initially set to zero by RTAB
S	Name of output vector length L+1 containing counts.
N	Number of rows in A.
M	Number of columns in A and R.
L	Number of rows in R.
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

Matrix R must be a general matrix.

Subroutines and Function Subprograms Required:

LOC
RADD

Method:

RTAB tabulates rows of data matrix A based on the key contained in vector B. It truncates the floating-point number in B(I) to form J. RTAB adds the Ith row of A to the Jth row of R element by element and 1 is added to S(J). If J is not between 1 and L, 1 is added to S(L+1).

This procedure is repeated for every element in vector B. Upon completion, the output matrix R contains a summary of row data as specified by vector B. Each element in vector S contains a count of the number of rows of A not included in R as a result of J being less than or greater than L.

3.2.34 RTIE Subroutine

RTIE adjoins two matrices with the same column dimension to form one resultant matrix (Test: MACHK2.ext).

Usage: CALL RTIE(A,B,R,N,M,MSA,MSB,L)

Parameter	Description
A	Name of first input matrix.
B	Name of second input matrix.
R	Name of output matrix.
N	Number of rows in A.
M	Number of columns in A, B, R.
MSA	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.
MSB	One-digit number for storage mode of matrix B: 0 General. 1 Symmetric. 2 Diagonal.
L	Number of rows in B.

Remarks:

Matrix R must be a general matrix and cannot be in the same location as either matrix A or matrix B. Matrix A must have the same number of columns as matrix B.

Subroutines and Function Subprograms Required:

LOC

Method:

RTIE attaches matrix B to the bottom of matrix A. The resultant matrix R contains N+L rows and M columns.

3.2.35 SADD Subroutine

SADD adds a scalar to each element of a matrix to form a resultant matrix (Test: MACHK4.ext).

Usage: CALL SADD(A,C,R,N,M,MS)

Parameter	Description
A	Name of input matrix.
C	Scalar.
R	Name of output matrix.
N	Number of rows in matrix A and R.
M	Number of columns in matrix A and R.
MS	One-digit number for storage mode of matrix A and R: 0 General. 1 Symmetric. 2 Diagonal.

Subroutines and Function Subprograms Required:

LOC

Method:

SADD adds a scalar to each element of the matrix.

3.2.36 SCLA Subroutine

SCLA sets each element of a matrix equal to a given scalar (Test: MACHK4.ext).

Usage: CALL SCLA(A,C,N,M,MS)

Parameter	Description
A	Name of input matrix.
C	Scalar.
N	Number of rows in matrix A.
M	Number of columns in matrix A.
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.

Subroutines and Function Subprograms Required:

LOC

Method:

SCLA replaces each element of matrix A by a scalar C.

3.2.37 SCMA Subroutine

SCMA multiplies a column of matrix by a scalar and adds the product to another column of the same matrix (Test: COLROW.ext).

Usage: CALL SCMA(A,C,N,LA,LB)

Parameter	Description
A	Name of matrix.
C	Scalar.
N	Number of rows in A.
LA	Column in A to be multiplied by a scalar.
LB	Column in A to which product is added. If 0 is specified, product replaces elements in LA.

Remarks:

Matrix A must be a general matrix.

Method:

SCMA multiplies each element of column LA by a scalar C and adds the product to the corresponding element of column LA. Column LA remains unaffected by the operation. If parameter LB contains zero, multiplication by the scalar is performed and the product replaces elements in LA.

3.2.38 SDIV Subroutine

SDIV divides each element of a matrix by a scalar to form a resultant matrix (Test: MACHK4.ext).

Usage: CALL SDIV(A,C,R,N,M,MS)

Parameter	Description
A	Name of input matrix.
C	Scalar.
R	Name of output matrix.
N	Number of rows in matrix A and R.
M	Number of columns in matrix A and R.
MS	One-digit number for storage mode of matrix A and R: 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

If the scalar is zero, SDIV divides only once to cause floating point overflow condition.

Subroutines and Function Subprograms Required:

LOC

Method:

SDIV divides each element of a matrix by a scalar.

3.2.39 SMPY Subroutine

SMPY multiplies each element of a matrix by a scalar to form a resultant matrix (Test: MACHK3.ext).

Usage: CALL SMPY(A,C,R,N,M,MS)

Parameter	Description
A	Name of input matrix.
C	Scalar.
R	Name of output matrix.
N	Number of rows in matrix A and R.
M	Number of columns in matrix A and R.
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.

Subroutines and Function Subprograms Required:

LOC

Method:

SMPY multiplies a scalar by each element of a matrix.

3.2.40 SRMA Subroutine

SRMA multiplies a row of matrix by a scalar and adds the product to another row of the same matrix (Test: COLROW.ext).

Usage: CALL SRMA(A,C,N,M,LA,LB)

Parameter	Description
A	Name of input matrix.
C	Scalar.
N	Number of rows in A.
M	Number of columns in A.
LA	Row in A to be multiplied by a scalar.

Parameter	Description
LB	Row in A to which the product is added. If 0 is specified, the product replaces elements in row LA.

Remarks:

Matrix A must be a general matrix.

Method:

SRMA multiplies each element of row LA by a scalar C and adds the product to the corresponding element of row LB. Row LA remains unaffected by the operation. If parameter LB contains zero, multiplication by the scalar is performed and the product replaces elements in row LA.

3.2.41 SSUB Subroutine

SSUB subtracts a scalar from each element of a matrix to form a resultant matrix (Test: MACHK4.ext).

Usage: CALL SSUB(A,C,R,N,M,MS)

Parameter	Description
A	Name of input matrix.
C	Scalar.
R	Name of output matrix.
N	Number of rows in matrix A and R.
M	Number of columns in matrix A and R.
MS	One-digit number for storage mode of matrix A and R: 0 General. 1 Symmetric. 2 Diagonal.

Subroutines and Function Subprograms Required:

LOC

Method:

SSUB subtracts a scalar from each element of a matrix.

3.2.42 TPRD Subroutine

TPRD transposes a matrix and postmultiplies it by another matrix to form a resultant matrix (Test: MACHK2.ext).

Usage: CALL TPRD(A,B,R,N,M,MSA,MSB,L)

Parameter	Description
A	Name of first input matrix.
B	Name of second input matrix.
R	Name of output matrix.
N	Number of rows in A and B.
M	Number of columns in A and rows in R.
MSA	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.
MSB	One-digit number for storage mode of matrix B: 0 General. 1 Symmetric. 2 Diagonal.
L	Number of columns in B and R.

Remarks:

Matrix R cannot be in the same location as either matrix A or matrix B.

Subroutines and Function Subprograms Required:

LOC

Method:

TPRD does not actually calculate the matrix transpose of A. Instead, it takes the elements in matrix A column-wise (rather than row-wise) for multiplication by matrix B.

The storage mode of the output matrix for all combinations of input matrices is as follows:

A	B	R
general	general	general
general	symmetric	general
general	diagonal	general
symmetric	general	general
symmetric	symmetric	general
symmetric	diagonal	general
diagonal	general	general
diagonal	symmetric	general
diagonal	diagonal	diagonal

3.2.43 XCPY Subroutine

XCPY copies a submatrix from a given matrix (Test: MACHK2.ext).

Usage: CALL XCPY(A,R,L,K,NR,MR,NA,MA,MS)

Parameter	Description
A	Name of input matrix.
R	Name of output matrix.
L	Row of A where first element of R can be found.
K	Column of A where first element of R can be found.
NR	Number of rows to be copied into R.
MR	Number of columns to be copied into R.
NA	Number of rows in A.
MA	Number of columns in A.
MS	One-digit number for storage mode of matrix A: 0 General. 1 Symmetric. 2 Diagonal.

Remarks:

Matrix R must be a general matrix and cannot be in the same location as matrix A.

Subroutines and Function Subprograms Required:

LOC

Method:

XCPY forms matrix R by copying a portion of matrix A. This is done by extracting NR rows and MR columns of matrix A, starting with element at row L, column K.

3.3 INTEGRATION AND DIFFERENTIATION SUBROUTINES

The sections that follow describe the integration and differentiation subroutines.

3.3.1 QATR Subroutine

QATR uses Romberg's extrapolation method to approximate the integral of a given function by trapezoidal rule (Test: INTEG.ext).

Usage: CALL QATR(XL,XU,EPS,NDIM,FCT,Y,IER,AUX)

Parameter	Description
XL	The lower bound of the interval.
XU	The upper bound of the interval.
EPS	The upper bound of the absolute error.

Parameter	Description
NDIM	The dimension of the auxiliary storage array AUX. NDIM-1 is the maximal number of bisections of the interval (XL,XU).
FCT	The name of the external function subprogram used.
Y	The resulting approximation for the integral value.
IER	Resultant error code: 0 It was possible to reach the required accuracy. No error. 1 It is impossible to reach the required accuracy because of rounding errors. 2 It was impossible to check accuracy because NDIM is less than 5, or the required accuracy could not be reached with NDIM-1 steps. NDIM should be increased.
AUX	An auxiliary storage array with dimension NDIM.

Remarks:

Parameter FCT requires an external statement.

Subroutines and Function Subprograms Required:

You must supply the external function subprogram FCT(X). Its argument X should not be destroyed.

Method:

QATR evaluates y by trapezoidal rule on Romberg's principle. On return, Y contains the best possible approximation of the integral value and vector AUX, the upward diagonal of Romberg's scheme. Components AUX(I) (I=1,2,...,IEND, with IEND less than or equal to NDIM) become approximations to integral value with decreasing accuracy by multiplication with (XU-XL). Refer to Filippi 1964 and Bauser 1961.

3.3.2 QSF Subroutine

QSF computes the vector of integral values for a given equidistant table of function values (Test: QDINT.ext).

Usage: CALL QSF(H,Y,Z NDIM)

Parameter	Description
H	The increment of argument values.
Y	The input vector of function values.
Z	The resulting vector of integral values. Z may be identical with Y.
NDIM	The dimension of vectors Y and Z.

Remarks:

No action is taken if NDIM is less than 3.

Method:

Beginning with Z(1)=0, evaluation of vector Z is done by means of Simpson's rule together with Newton's 3/8 rule or a combination of these two rules. Truncation error is of the order H**5 (that is, the fourth order method). Only in case NDIM=3 truncation error of Z(2) is of order H**4. Refer to Hildebrand 1974 and Zurmuehl 1963.

3.3.3 RKGS Subroutine

RKGS solves a system of first-order ordinary differential equations with initial values by the Runge-Kutta method (Test: RKGSTT.ext).

Usage: CALL RKGS(PRMT,Y,DERY,NDIM,IHLF,FCT,OUTP,AUX)

Parameter	Description
PRMT	An input and output vector with dimension greater than or equal to 5, that specifies the parameters of the interval and of accuracy, and serves for communication between output subroutine (furnished by you) and subroutine RKGS. Except PRMT(5), the components are not destroyed by RKGS. These components are:
PRMT(1)	Lower bound of the interval (input).
PRMT(2)	Upper bound of the interval (input).
PRMT(3)	Initial increment of the independent variable (input).
PRMT(4)	Upper error bound (input). If absolute error is greater than PRMT(4), increment gets halved. If increment is less than PRMT(4)/50, increment gets doubled. You may change PRMT(4) by means of your output subroutine.
PRMT(5)	No input parameter. RKGS initializes PRMT(5)=0. If you want to terminate subroutine RKGS at any point, you must change PRMT(5) to nonzero by means of subroutine OUTP. Further components of vector PRMT are feasible if its dimension is defined greater than 5; however, subroutine RKGS does not require and change them. Nevertheless, they can be useful for handing result values to the main program (calling RKGS) that are obtained by special manipulations with output data in subroutine OUTP.
Y	Input vector of initial values (destroyed). Later, Y is the resulting vector of dependent variables computed at intermediate points X.
DERY	Input vector of error weights (destroyed). The sum of its components must be equal to 1. Later, DERY is the vector of derivatives that belong to function values Y at a point X.

Parameter	Description
NDIM	An input value that specifies the number of equations in the system.
IHLF	An output value that specifies the number of bisections of the initial increment. If IHLF becomes greater than 10, subroutine RKGS returns with error message IHLF=11 into main program. Error message IHLF=12 or IHLF=13 appears in case PRMT(3)=0 or in case SIGN(PRMT(3)).NE.SIGN(PRMT(2)-PRMT(1)), respectively.
FCT	The name of an external subroutine used. This subroutine computes the right-hand side DERY of the system to given values X and Y. Its parameter list must be X,Y,DERY. Subroutine FCT should not destroy X and Y.
OUTP	The name of an external output subroutine used. Its parameter list must be X,Y,DERY,IHLF,NDIM,PRMT. None of these parameters (except, if necessary, PRMT(4), PRMT(5),...) should be changed by subroutine OUTP. If PRMT(5) is changed to nonzero, subroutine RKGS is terminated.
AUX	An auxiliary storage array with eight rows and NDIM columns.

Remarks:

The procedure terminates and returns to calling program if:

1. More than 10 bisections of the initial increment are necessary for satisfactory accuracy (error message IHLF=11).
2. Initial increment is equal to 0 or has wrong sign (error messages IHLF=12 or IHLF=13).
3. The whole integration interval is worked through.
4. Subroutine OUTP has changed PRMT(5) to nonzero.

Subroutines and Function Subprograms Required:

You must supply the external subroutines FCT(X,Y,DERY) and OUTP(X,Y,DERY,IHLF,NDIM,PRMT).

Method:

Evaluation is done by means of fourth order Runge-Kutta formula in the modification due to Gill. Accuracy is tested comparing the results of the procedure with single and double increment.

RKGS automatically adjusts the increment during the whole computation by halving or doubling. If more than 10 bisections of the increment are necessary to get satisfactory accuracy, RKGS returns with the error message IHLF=11 into the main program.

You must furnish an output subroutine to get full flexibility in output. Refer to Ralston and Wilf, eds. 1960.

3.3.4 RK1 Subroutine

RK1 integrates a first-order differential equation dY/dX=FUN(X,Y) up to a specified final value (Test: INTEG.ext).

Usage: CALL RK1(FUN,HI,XI,YI,XF,YF,ANSX,ANSY,IER)

Parameter	Description
FUN	User-supplied function subprogram with arguments X,Y that give dY/dX.
HI	The step size.
XI	Initial value of X.
YI	Initial value of Y where YI=Y(XI).
XF	Final value of X.
YF	Final value of Y.
ANSX	Resultant final value of X.
ANSY	Resultant final value of Y. Either ANSX will equal XF or ANSY will equal YF, depending on which is reached first.
IER	Resultant error code: 0 No error. 1 Step size is zero.

Remarks:

If XI is greater than XF, ANSX=XI and ANSY=YI. If HI IS zero, IER is set to 1, ANSX is set to XI, and ANSY is set to zero.

Subroutines and Function Subprograms Required:

FUN is a two-argument function subprogram. You must furnish FUN (dY/dX=FUN(X,Y)).

Your calling program must have a FORTRAN external statement containing the names of the function subprograms listed in the call to RK1.

Method:

RK1 uses fourth order Runge-Kutta integration process on a recursive basis (Hildebrand 1974). Process is terminated and final value adjusted when either XF or YF is reached.

3.3.5 RK2 Subroutine

RK2 integrates a first-order differential equation dY/dX=FUN(X,Y) by Runge-Kutta and produces a table of the integrated values (Test: RK2INT.ext).

Usage: CALL RK2(FUN,H,XI,YI,K,N,VEC)

Parameter	Description
FUN	User-supplied function subprogram with arguments X,Y that give dY/dX.
H	Step size.
XI	Initial value of X.
YI	Initial value of Y where YI=Y(XI).
K	The interval at which computed values are to be stored.
N	The number of values to be stored.
VEC	The resultant vector of length N, in which computed values of Y are to be stored.

Subroutines and Function Subprograms Required:

FUN is a subprogram you must supply for dY/dX. Your calling program must have a FORTRAN external statement containing the names of function subprograms listed in the call to RK2.

Method:

RK2 uses fourth-order Runge-Kutta integration on a recursive basis (Hildebrand 1974).

3.4 FOURIER ANALYSIS SUBROUTINES

3.4.1 FORIF Subroutine

FORIF computes the coefficients of the desired number of terms in the Fourier Series F(X)=A(0)+SUM(A(K)COS KX+B(K)SIN KX) where K=1,2,...,M to approximate the computed values of a given function subprogram (Test: FORR.ext).

Usage: CALL FORIF(FUN,N,M,A,B,IER)

Parameter	Description
FUN	Name of function subprogram to be used for computing data points.
N	Defines the interval such that 2N+1 points are taken over the interval (0,2PI). The spacing is thus 2PI/2N.
M	The maximum order of the harmonics to be fitted.
A	Resultant vector of Fourier cosine coefficients of length M+1. A SUB 0, A SUB 1, ..., A SUB M
B	Resultant vector of Fourier sine coefficients of length M+1. B SUB 0, B SUB 1, ..., B SUB M

Parameter	Description
IER	Resultant error code: 0 No Error. 1 N not greater than or equal to M. 2 M less than 0.

Remarks:

M must be greater than or equal to zero. N must be greater than or equal to M. The first element in vector B is zero in all cases.

Subroutines and Function Subprograms Required:

FUN is the name of a user function subprogram that computes data points. The calling program must have a FORTRAN external statement containing names of function subprograms listed in a call to FORIF.

Method:

FORIF uses the recursive technique described in Ralston and Wilf, eds. 1960. The method of indexing through the procedure has been modified to simplify the computation.

3.4.2 FORIT Subroutine

FORIT computes the coefficients of a specified number of terms in the Fourier Series to approximate a given set of periodically tabulated values of a function (Test: FORR.ext).

Usage: CALL FORIT(FOR,N,M,A,B,IER)

Parameter	Description
FOR	Vector of tabulated function values of length 2N+1.
N	Defines the interval such than 2N+1 points are taken over the interval (0,2PI). The spacing is thus 2PI/2N.
M	Maximum order of harmonics to be fitted.
A	Resultant vector of Fourier cosine coefficients of length M+1. A SUB 0, A SUB 1, ..., A SUB M
B	Resultant vector of Fourier sine coefficients of length M+1. B SUB 0, B SUB 1, ..., B SUB M
IER	Resultant error code: 0 No error. 1 N not greater than or equal to M. 2 M less than 0.

Remarks:

M must be greater than or equal to 0. N must be greater than or equal to M. The first element of vector B is 0 in all cases.

Method:

FORIT uses the recursive technique described in Ralston and Wilf, eds., 1960. The method of indexing through the procedure has been modified to simplify the computation.

3.5 SPECIAL SUBROUTINE OPERATIONS AND FUNCTIONS

3.5.1 BESI Subroutine

BESI computes the I Bessel function for a given argument and order using series or asymptotic approximation (Test: TABLE2.ext).

Usage: CALL BESI(X,N,BI,IER)

Parameter	Description
X	The argument of the I Bessel function desired.
N	The order of the I Bessel function desired.
BI	The resultant I Bessel function.
IER	Resultant error code: 0 No error. 1 N is negative. 2 X is negative. 3 Underflow, BI.LT.1.E-38, BI set to 0.0. 4 Overflow, X.GT.60 where X.GT.N.

Remarks:

N and X must be greater than or equal to zero.

Method:

BESI computes the I Bessel function using series or asymptotic approximation depending on range of arguments.

3.5.2 BESJ Subroutine

BESJ computes the J Bessel function for a given argument and order using recurrence relation technique (Test: TABLE2.ext).

Usage: CALL BESJ(X,N,BJ,D,IER)

Parameter	Description
X	The argument of the J Bessel function desired.
N	The order of the J Bessel function desired.
BJ	The resultant J Bessel function.
D	Required accuracy.
IER	Resultant error code: 0 No error. 1 N is negative. 2 X is negative or zero. 3 Required accuracy not obtained. 4 Range of N compared to X not correct (see Remarks).

Remarks:

The value of N must be greater than or equal to zero, but it must be less than:

```
20+10*X-X**2/3      for X less than or equal to 15
90+(X/2)            for X greater than 15
```

Method:

BESJ uses the recurrence relation technique described in Goldstein and Thaler, and Stegun and Abramowitz 1957.

3.5.3 BESK Subroutine

BESK computes the K Bessel function for a given argument and order using series approximations and recurrence relations (Test: TABLE2.ext).

Usage: CALL BESK(X,N,BK,IER)

Parameter	Description
X	The argument of the K Bessel function desired.
N	The order of the K Bessel function desired.
BK	The resultant K Bessel function.
IER	Resultant error code: 0 No error. 1 N is negative. 2 X is zero or negative. 3 X.GT.60, machine range exceeded. 4 BK.GT.10**36.

Remarks:

The value of N must be greater than or equal to zero.

Method:

BESK computes zero-order and first-order Bessel functions using series approximations and then computes Nth order function using the recurrence-relation and polynomial-approximation technique, as described in Hitchcock 1957 and Watson 1958.

3.5.4 BESY Subroutine

BESY computes the Y Bessel function for a given argument and order using recurrence relations and polynomial approximations (Test: TABLE2.ext).

Usage: CALL BESY(X,N,BY,IER)

Parameter	Description
X	The argument of the Y Bessel function desired.
N	The order of the Y Bessel function desired.
BY	The resultant Y Bessel function.
IER	Resultant error code: 0 No error. 1 N is negative. 2 X is negative or zero. 3 BY has exceeded magnitude of 10**36.

Remarks:

Very small values of X can cause the range of the library function ALOG to be exceeded. X must be greater than zero. N must be greater than or equal to zero.

Method:

BESY uses recurrence-relation and polynomial-approximation technique as described in Hitchcock 1957 and Watson 1958.

3.5.5 CEL1 Subroutine

CEL1 computes the complete elliptic integral of the first kind using Landens transformation (Test: TABLE3.ext).

Usage: CALL CEL1(RES,AK,IER)

Parameter	Description
RES	Result value.
AK	Modulus (input).
IER	Resultant error code: 0 No error. 1 AK not in range -1 to +1.

Remarks:

If ABS(AK) is greater than or equal to 1, CEL1 sets the result to 1.E38. For modulus AK and complementary modulus CK, equation AK*AK+CK*CK=1.0 is used. AK must be in the range -1 to +1.

Method:

CEL1 uses Landen's transformation for calculation (Bulirsch 1965).

3.5.6 CEL2 Subroutine

CEL2 computes the generalized complete elliptic integral of the second kind (Test: TABLE3.ext).

Usage: CALL CEL2(RES,AK,A,B,IER)

Parameter	Description
RES	Result value.
AK	Modulus (input).
A	Constant term in numerator.
B	Factor of quadratic term in numerator.
IER	Resultant error code: 0 No error. 1 AK not in range -1 to +1.

Remarks:

For ABS(AK).GE.1 CEL2 sets the result to 1.E38 if B is positive, to -1.E38 if B is negative. Special cases are:

K(K)	obtained with A=1, B=1.
E(K)	obtained with A=1, B=CK*CK where CK is the complementary modulus.
B(K)	obtained with A=1, B=0.
D(K)	obtained with A=0, B=1.

Where K, E, B, and D define special cases of the generalized complete elliptic integral of the second kind in the usual notation, and the argument K of these functions is the modulus.

Method:

CEL2 uses Landen's transformation for calculation (Bulirsch 1965).

3.5.7 CS Subroutine

CS computes the Fresnel integrals using rational function approximations (Test: TABLE1.ext).

Usage: CALL CS(C,S,X)

Parameter	Description
C	The resultant value C(X).
S	The resultant value S(X).
X	The argument of Fresnel integrals. If X is negative, the absolute value is used.

Remarks:

The argument value X remains unchanged.

Method:

Refer to Boersma 1960.

3.5.8 EXPI Subroutine

EXPI computes the exponential integral -EI(-X) using three different rational approximations (Test: TABLE3.ext).

Usage: CALL EXPI(X,RES,AUX)

Parameter	Description
X	Argument of exponential integral.
RES	Result value.
AUX	Resultant auxiliary value.

Remarks:

A value of X greater than 87 can cause overflow with the exponential function. For X=0, EXPI sets the result to -1.E37.

Method:

EXPI uses three rational approximations in the ranges 1 less than or equal to X, X less than or equal to -9, and -9 less than X less than or equal to -3 respectively. It uses polynomial approximation in -3 less than X less than 1.

3.5.9 GAMMA Subroutine

GAMMA computes the GAMMA function for a given argument using the recursion relation and polynomial approximation (Test: TABLE1.ext).

Usage: CALL GAMMA(XX,GX,IER)

Parameter	Description
XX	The argument for the Gamma function.
GX	The resultant Gamma function value.

Parameter	Description
IER	Resultant error code: 0 No error. 1 XX is within .000001 of being a negative integer. 2 XX.GT.34.5, overflow, GX set to 1.0E38.

Method:

Refer to Hastings 1955.

3.5.10 LEP Subroutine

LEP computes the values of the Legendre polynomials P(N,X) for argument value X and orders 0 to N (Test: TABLE1.ext).

Usage: CALL LEP(Y,X,N)

Parameter	Description
Y	Result vector of dimension N+1 containing the values of Legendre polynomials of order 0 up to N for given argument X.
X	Argument of Legendre polynomial.
N	Order of Legendre polynomial.

Remarks:

LEP treats values of N less than zero as zero.

Method:

The evaluation method used by LEP is based on the recurrence equation for Legendre polynomials:

$$P(N+1,X) = 2*X*P(N,X)-P(N-1,X)-(X*P(N,X)-P(N-1,X))/(N+1)$$

The first term in brackets is the order. The second is the argument. Starting values are:

$$P(0,X)=1,\ P(1,X)=X$$

3.5.11 SICI Subroutine

SICI computes the sine and cosine integrals where SI(x)=integral(Sin(X)/X)-PI/2, and CI(x)=integral(Cos(X)/X) (Test: TABLE3.ext).

Usage: CALL SICI(SI,CI,X)

Parameter	Description
SI	The resultant value SI(X).
CI	The resultant value CI(X).
X	The argument of SI(X) and CI(X).

Remarks:

The argument value remains unchanged.

Method:

Refer to Luke and Wimp 1961.

3.6 LINEAR EQUATIONS SUBROUTINE

The section that follows describes the linear equations subroutine.

3.6.1 SIMQ Subroutine

SIMQ solves a set of simultaneous linear equations, AX=B (Test: SOLVEN.ext).

Usage: CALL SIMQ(A,B,N,KS)

Parameter	Description
A	Matrix of coefficients stored in columns. These are destroyed in the computation. The size of matrix A is N by N.
B	Vector of original constants (length N). These are replaced by final solution values, vector X.
N	Number of equations and variables. N must be greater than 1.
KS	Output digit: 0 for a normal solution. 1 for a singular set of equations.

Remarks:

Matrix A must be general. If the matrix is singular, the solution values are meaningless. You may obtain an alternative solution by using matrix inversion (MINV) and matrix product (GMPRD).

Method:

The method of solution is by elimination, using the largest pivotal divisor. Each stage of elimination consists of interchanging rows when necessary to avoid division by zero or small elements.

The forward solution to obtain variable N is done in N stages. The back solution for the other variables is calculated by successive substitutions.

Final solution values are developed in vector B, with variable 1 in B(1), variable 2 in B(2) , ..., and variable N in B(N). If no pivot can be found exceeding a tolerance of 0.0, the matrix is considered singular, and KS is set to 1. This tolerance can be modified by replacing the first statement.

3.7 NONLINEAR EQUATIONS SUBROUTINES

The sections that follow describe the nonlinear equations subroutines.

3.7.1 RTMI Subroutine

RTMI solves the general nonlinear equation of the form FCT(X)=0 using Mueller's iteration method (Test: NONLIN.ext).

Usage: CALL RTMI(X,F,FCT,XLI,XRI,EPS,IEND,IER)

Parameter	Description
X	Resultant root of equation FCT(X)=0.
F	Resultant function value at root X.
FCT	Name of the external function subprogram used.
XLI	Input value that specifies the initial left bound of the root X.
XRI	Input value that specifies the initial right bound of the root X.
EPS	Input value that specifies the upper bound of the error of result X.
IEND	Maximum number of iteration steps specified.
IER	Resultant error code: 0 No error. 1 No convergence after IEND iteration steps followed by IEND successive steps of bisection. 2 Basic assumption FCT(XLI)*FCT(XRI) less than or equal to zero is not satisfied.

Remarks:

Parameter FCT requires a FORTRAN external statement. The procedure assumes that function values at initial bounds XLI and XRI do not have the same sign. If this basic assumption is not satisfied by input values XLI and XRI, RTMl bypasses the procedure and gives the error message IER=2.

Subroutines and Function Subprograms Required:

You must supply the external function subprogram FCT(X).

Method:

The solution of the equation FCT(X)=0 is by Mueller's iteration method of successive bisections and inverse parabolic interpolation, that starts at the initial bounds XLI and XRI. Convergence is quadratic if the derivative of FCT(X) at root X is not equal to zero. One iteration step requires two evaluations of FCT(X). Refer to Kristiansen 1963.

3.7.2 RTNI Subroutine

RTNI solves the general nonlinear equation of the form FCN(X)=0 by Newton's iteration method (Test: NONLIN.ext).

Usage: CALL RTNI(X,F,DERF,FCT,XST,EPS,IEND,IER)

Parameter	Description
X	Resultant root of equation F(X)=0.
F	Resultant function value at root X.
DERF	Resultant value of derivative at root X.
FCT	Name of external subroutine used. It computes to given argument X, function value F, and derivative DERF. Its parameter list must be X, F, DERF.
XST	Input value that specifies the initial guess of the root X.
EPS	Input value that specifies the upper bound of the error of result X.
IEND	Maximum number of iteration steps specified.
IER	Resultant error code: 0 No error. 1 No convergence after IEND iteration steps. 2 At any iteration step derivative DERF was equal to zero.

Remarks:

The parameter FCT requires a FORTRAN external statement. The procedure is bypassed and gives the error message IER=2 if at any iteration step the derivative of F(X) is equal to 0. Possibly the procedure would be successful if it started once more with another initial guess XST.

Subroutines and Function Subprograms Required:

You must supply the external subroutine FCT(X,F,DERF).

Method:

Solution of equation F(X)=0 is by Newton's iteration method, that starts at the initial guess XST of a root X. Convergence is quadratic if the derivative of F(X) at root X is not equal to zero. One iteration step requires one evaluation of F(X) and one evaluation of the derivative of F(X). Refer to Zurmuehl 1963.

3.7.3 RTWI Subroutine

RTWI solves the general nonlinear equation of the form FCT(X)=X by Wegstein's iteration method (Test: NONLIN.ext).

Usage: CALL RTWI(X,VAL,FCT,XST,EPS,IEND,IER)

Parameter	Description
X	Resultant root of equation X=FCT(X).
VAL	Resultant value of X=FCT(X) at root X.
FCT	Name of external function subprogram used.
XST	Input value that specifies the initial guess of the root X.
EPS	Input value that specifies the upper bound of the error of result X.
IEND	Maximum number of iteration steps specified.
IER	Resultant error code: 0 No error. 1 No convergence after IEND iteration steps. 2 At any iteration step the denominator of iteration formula was equal to zero.

Remarks:

The parameter FCT requires an external statement. RTWI bypasses the procedure and gives the error message IER=2 if at any iteration step the denominator of iteration formula is equal to zero. That means that there is at least one point in the range in which iteration occurs with derivative of FCT(X)=1.

Subroutines and Function Subprograms Required:

You must supply the external function subprogram FCT(X).

Method:

RTWI solves the equation X=FCT(X) by Wegstein's iteration method, that starts at the initial guess XST of root X. One iteration step requires one evaluation of FCT(X). Refer to Lance 1960, Wegstein 1960, Thacher 1960, and Herriot 1960.

3.8 ROOTS OF POLYNOMIALS SUBROUTINE

The section that follows describes the roots of polynomials subroutine.

3.8.1 POLRT Subroutine

POLRT finds real and complex roots of a real polynomial using the Newton-Raphson iterative technique (Test: SMPRT.ext).

Usage: CALL POLRT(XCOF,COF,M,ROOTR,ROOTI,IER)

Parameter	Description
XCOF	Vector of M+1 coefficients of the polynomial ordered from smallest to largest power.
COF	Working vector of length M+1.
M	Order of polynomial.
ROOTR	Resultant vector of length M containing real roots of the polynomial.
ROOTI	Resultant vector of length M containing the corresponding imaginary roots of the polynomial.
IER	Resultant error code: 0 No error. 1 M less than 1. 2 M greater than 36. 3 Unable to determine root with 500 iterations on 5 starting values. 4 High-order coefficient is 0.

Remarks:

POLRT is limited to 36th order polynomial or less. Floating-point overflow can occur, but is accommodated by the subroutine and does not affect the results.

Method:

POLRT uses the Newton-Raphson iterative technique. To avoid errors in the reduced polynomial, it performs the final iterations on each root using the original polynomial rather than the reduced polynomial.

3.9 POLYNOMIAL OPERATIONS SUBROUTINES

The sections that follow describe the polynomial operations subroutines.

3.9.1 PADD Subroutine

PADD adds two polynomials (Test: POLY2.ext).

Usage: CALL PADD(Z,IDIMZ,X,IDIMX,Y,IDIMY)

Parameter	Description
Z	Vector of resultant coefficients, ordered from smallest to largest power.
IDIMZ	Dimension of Z (calculated).
X	Vector of coefficients for first polynomial, ordered from smallest to largest power.
IDIMX	Dimension of X (degree is IDIMX-1).

Parameter	Description
Y	Vector of coefficients for second polynomial, ordered from smallest to largest power.
IDIMY	Dimension of Y (degree is IDIMY-1).

Remarks:

Vector Z can be in the same location as either vector X or vector Y only if the dimension of Z is not less than the other input vector. The resultant polynomial can have trailing zero coefficients.

Method:

PADD calculates the dimension of resultant vector IDIMZ as the larger of the two input vector dimensions. Then, it adds corresponding coefficients to form Z.

3.9.2 PADDM Subroutine

PADDM multiplies a polynomial by a constant and adds the result to another polynomial (Tests: POLY1.ext, POLY2.ext).

Usage: CALL PADDM(Z,IDIMZ,X,IDIMX,FACT,Y,IDIMY)

Parameter	Description
Z	Vector of resultant coefficients, ordered from smallest to largest power.
IDIMZ	Dimension of Z (calculated).
X	Vector of coefficients for first polynomial, ordered from smallest to largest power.
IDIMX	Dimension of X (degree is IDIMX-1).
FACT	Factor to be multiplied by vector Y.
Y	Vector of coefficients for second polynomial, ordered from smallest to largest power.
IDIMY	Dimension of Y (degree is IDIMY-1).

Remarks:

Vector Z can be in the same location as either vector X or vector Y only if the dimension of Z is not less than the other input vector. The resultant polynomial can have trailing zero coefficients.

Method:

PADDM calculates the dimension of resultant vector IDIMZ as the larger of the two input vector dimensions. Then, PADDM adds the coefficient in vector X to the coefficient in vector Y multiplied by factor to form Z.

3.9.3 PCLA Subroutine

PCLA replaces one polynomial by another (Test: POLY2.ext).

Usage: CALL PCLA(Y,IDIMY,X,IDIMX)

Parameter	Description
Y	Vector of resultant coefficients, ordered from smallest to largest power.
IDIMY	Dimension of Y.
X	Vector of coefficients for polynomial, ordered from smallest to largest power.
IDIMX	Dimension of X.

Method:

PCLA replaces IDIMY by IDIMX and moves vector X to Y.

3.9.4 PCLD Subroutine

PCLD performs complete linear synthetic division (shift of origin) (Test: POLY2.ext).

Usage: CALL PCLD(X,IDIMX,U)

Parameter	Description
X	Vector of coefficients, ordered from smallest to largest power. It is replaced by vector of transformed coefficients.
IDIMX	Dimension of X.
U	Shift parameter.

Method:

PCLD transforms coefficient vector X(I) of polynomial P(Z) so that Q(Z)=P(Z-U) (where Q(Z) denotes the polynomial with transformed coefficient vector).

3.9.5 PDER Subroutine

PDER finds the derivative of a polynomial (Test: POLY1.ext).

Usage: CALL PDER(Y,IDIMY,X,IDIMX)

Parameter	Description
Y	Vector of coefficients for derivative, ordered from smallest to largest power.
IDIMY	Dimension of Y (equal to IDIMX-1).

Parameter	Description
X	Vector of coefficients for original polynomial, ordered from smallest to largest power.
IDIMX	Dimension of X.

Method:

PDER sets the dimension of Y at the dimension of X-1. Then, it calculates the derivative by multiplying the coefficients by their respective exponents.

3.9.6 PDIV Subroutine

PDIV divides one polynomial by another (Test: POLY1.ext).

Usage: CALL PDIV(P,IDIMP,X,IDIMX,Y,IDIMY,TOL,IER)

Parameter	Description
P	Resultant vector of integral part.
IDIMP	Dimension of P.
X	Vector of coefficients for dividend polynomial, ordered from smallest to largest power. It is replaced by remainder after division.
IDIMX	Dimension of X.
Y	Vector of coefficients for divisor polynomial, ordered from smallest to largest power.
IDIMY	Dimension of Y.
TOL	Tolerance value below which coefficients are eliminated during normalization.
IER	Resultant error code: 0 No error. 1 Divisor is 0.

Remarks:

The remainder R replaces X. The divisor Y remains unchanged. If the dimension of Y exceeds dimension of X, PDIV sets IDIMP to zero and bypasses calculation.

Subroutines and Function Subprograms Required:

PNORM

Method:

PDIV divides polynomial X by polynomial Y giving the integer part P and one remainder R so that X=P*Y+R. PDIV normalizes divisor Y and remainder vector.

3.9.7 PGCD Subroutine

PGCD finds greatest common divisor of two polynomials (Test: POLY2.ext).

Usage: CALL PGCD(X,IDIMX,Y,IDIMY,WORK,EPS,IER)

Parameter	Description
X	Vector of coefficients for first polynomial, ordered from smallest to largest power.
IDIMX	Dimension of X.
Y	Vector of coefficients for second polynomial, ordered from smallest to largest power. This is replaced by greatest common divisor.
IDIMY	Dimension of Y.
WORK	Working storage array.
EPS	Tolerance value below which coefficient is eliminated during normalization.
IER	Resultant error code: 0 means no error. 1 X or Y is zero polynomial.

Remarks:

IDIMX must be greater than IDIMY. If IDIMY=1 on return to the calling program, then X and Y are prime and the greatest common divisor is constant. IDIMX is destroyed during compilation.

Subroutines and Function Subprograms Required:

PDIV
PNORM

Method:

PGCD determines the greatest common divisor of two polynomials X and Y by the Euclidean algorithm. Coefficient vectors X and Y are destroyed and greatest common divisor is generated in Y.

3.9.8 PILD Subroutine

PILD evaluates a polynomial and its first derivative for a given argument (Test: POLY2.ext).

Usage: CALL PILD(POLY,DVAL,ARGUM,X,IDIMX)

Parameter	Description
POLY	Value of polynomial.
DVAL	Derivative.
ARGUM	Argument.

Parameter	Description
X	Vector of coefficients for polynomial, ordered from smallest to largest power.
IDIMX	Dimension of X.

Subroutines and Function Subprograms Required:

PQSD

Method:

PILD uses the subroutine PQSD (quadratic synthetic division) to evaluate the polynomial.

3.9.9 PINT Subroutine

PINT finds the integral of a polynomial with a constant of integration equal to zero (Test: POLY1.ext).

Usage: CALL PINT(Y,IDIMY,X,IDIMX)

Parameter	Description
Y	Vector of coefficients for integral, ordered from smallest to largest power.
IDIMY	Dimension of Y (equal to IDIMX+1).
X	Vector of coefficients for original polynomial, ordered from smallest to largest power.
IDIMX	Dimension of X.

Method:

PINT sets the dimension of Y at dimension of X+1, and sets the constant term to zero. PINT then calculates the integral by dividing the coefficients by their respective exponents.

3.9.10 PMPY Subroutine

PMPY multiplies two polynomials (Tests: POLY1.ext, POLY2.ext).

Usage: CALL PMPY(Z,IDIMZ,X,IDIMX,Y,IDIMY)

Parameter	Description
Z	Vector of resultant coefficients, ordered from smallest to largest power.
IDIMZ	Dimension of Z (calculated).
X	Vector of coefficients for first polynomial, ordered from smallest to largest power.
IDIMX	Dimension of X (degree is IDIMX-1).

Parameter	Description
Y	Vector of coefficients for second polynomial, ordered from smallest to largest power.
IDIMY	Dimension of Y (degree is IDIMY-1).

Remarks:

Vector Z cannot be in the same location as either vector X or vector Y.

Method:

PMPY calculates the dimension of Z as IDIMX+IDIMY-1. PMPY calculates the coefficients of Z as the sum of the products of coefficients of X and Y, whose exponents equal the corresponding exponent of Z.

3.9.11 PNORM Subroutine

PNORM normalizes coefficient vector of a polynomial (Test: POLY1.ext).

Usage: CALL PNORM(X,IDIMX,EPS)

Parameter	Description
X	Vector of original coefficients, ordered from smallest to largest power. It remains unchanged.
IDIMX	Dimension of X. It is replaced by final dimension.
EPS	Tolerance below which coefficient is eliminated.

Remarks:

If all coefficients are less than EPS, the result is zero polynomial with IDIMX=0. The vector X remains intact.

Method:

PNORM reduces the dimension of vector X by 1 for each trailing coefficient with an absolute value less than or equal to EPS.

3.9.12 PQSD Subroutine

PQSD performs quadratic synthetic division of a polynomial (Test: POLY2.ext).

Usage: CALL PQSD(A,B,P,Q,X,IDIMX)

Parameter	Description
A	Coefficient of Z in remainder (calculated).
B	Constant term in remainder (calculated).
P	Coefficient of Z in quadratic polynomial.
Q	Constant term in quadratic polynomial.

Parameter	Description
X	Coefficient vector for given polynomial, ordered from smallest to largest power.
IDIMX	Dimension of X.

Method:

PQSD divides polynomial by the quadratic Z**2-P*Z-Q, giving the linear remainder A*Z+B.

3.9.13 PSUB Subroutine

PSUB subtracts one polynomial from another (Test: POLY2.ext).

Usage: CALL PSUB(Z,IDIMZ,X,IDIMX,Y,IDIMY)

Parameter	Description
Z	Vector of resultant coefficients, ordered from smallest to largest power.
IDIMZ	Dimension of Z (calculated).
X	Vector of coefficients for first polynomial, ordered from smallest to largest power.
IDIMX	Dimension of X (degree is IDIMX-1).
Y	Vector of coefficients for second polynomial, ordered from smallest to largest power.
IDIMY	Dimension of Y (degree is IDIMY-1).

Remarks:

Vector Z may be in same location as either vector X or vector Y only if the dimension of X is not less than the other input vector. The resultant polynomial can have trailing zero coefficients.

Method:

PSUB calculates the dimension of resultant vector IDIMZ as the larger of the two input vector dimensions. It then subtracts coefficients in vector Y from corresponding coefficients in vector X.

3.9.14 PVAL Subroutine

PVAL evaluates a polynomial for a given value of the variable (Test: POLY1.ext).

Usage: CALL PVAL(RES,ARG,X,IDIMX)

Parameter	Description
RES	Resultant value of polynomial.
ARG	Given value of variable.
X	Vector of coefficients, ordered from smallest to largest power.
IDIMX	Dimension of X.

Method:

Evaluation is done by means of nested multiplication.

3.9.15 PVSUB Subroutine

PVSUB substitutes a polynomial for the variable of another polynomial (Test: POLY2.ext).

Usage: CALL PVSUB(Z,IDIMZ,X,IDIMX,Y,IDIMY,WORK1,WORK2)

Parameter	Description
Z	Vector of coefficients for resultant polynomial, ordered from smallest to largest power.
IDIMZ	Dimension of Z.
X	Vector of coefficients for original polynomial, ordered from smallest to largest power.
IDIMX	Dimension of X.
Y	Vector of coefficients for the polynomial that is substituted for the variable, ordered from smallest to largest power.
IDIMY	Dimension of Y.
WORK1	Working storage array (same dimension as Z).
WORK2	Working storage array (same dimension as Z).

Subroutines and Function Subprograms Required:

PMPY
PADDM
PCLA

Method:

PVSUB substitutes polynomial Y for the variable of polynomial X to form polynomial Z. The dimension of the new polynomial is (IDIMX-1)*(IDIMY-1)+1. PVSUB requires two work areas.

3.10 REFERENCES

Bauser. "Algorithm 60," CACM, Vol. 4, Iss, 6 (1961), p. 255.

Boersma. "Computation of Fresnel Integrals," Mathematical Tables and Other Aids to Computation, Vol. 14, No. 72 (1960), p. 380.

Bulirsch, R. "Numerical Calculation of Elliptic Integrals and Elliptic Functions," Handbook Series Special Functions, Numerische Mathematik, Vol. 7 (1965), pp. 78-90.

Filippi. "Das Verfahrem von Romberg-Stiefel-Bauer als Spezialfall des Allgemeinen Prinzips von Richardson," Mathematik-Technik-Wirtschaft, Vol. 11, Iss. 2 (1964), pp. 49-54.

Golstein, H., and R.M. Thaler. "Recurrence Techniques for the Calculation of Bessel Functions," M.T.A.C., Vol. 13, pp. 102-108.

Hasting, C. Jr. "The Recursion Relation and Polynomial Approximation," Approximation for Digital Computers. 1955.

Herriot, J.G. "Algorithm 26," CACM, Vol. 3, Iss .11 (1960), p. 603.

Hildebrand, F.B. Introduction to Numerical Analysis. New York: McGraw-Hill, 1974.

Hitchcock, A.J.M. "Polynomial Approximations to Bessel Functions of Order Zero and One and Two Related Functions," M.T.A.C., Vol. 11 (1957), pp. 86-88.

Kristiansen, G.K. "Zero of Arbitrary Function," Bit, Vol. 3 (1963), pp.205-206.

Lance, G.N. Numerical Methods for High Speed Computers. London: Iliffe, 1960.

Luke, and Wimp. "Polynomial Approximation to Integral Transforms," Mathematical Tables and Other Aids to Computation, Vol. 15, No. 74 (1961), pp. 174-178.

Ralston, A., and H.S. Wilf, eds. Mathematical Methods for Digital Computers. New York: John Wiley and Sons, 1962.

Stegun, I.A., and M. Abramowitz. "Generation of Bessel Functions on High Speed Computers," M.T.A.C., Vol. 11 (1957), pp. 255-257.

Thatcher, H.C. "Algorithm 15," CACM, Vol. 3, Iss. 8 (1960), p. 475.

Watson, G.N. A Treatise on the Theory of Bessel Functions. London: Cambridge University Press, 1958.

Wegstein, J. "Algorithm 2," CACM, Vol. 3, Iss. 8 (1960), p. 74.

Zurmuehl, R. Praktische Mathematik fuer Ingenieure und Physiker. Springer, 1963.

APPENDIX A

INSTALLING, VERIFYING, AND USING SSP UNDER RT-11

This appendix explains how to install and verify the Scientific Subroutines Package (SSP) software and how to create a FORTRAN IV program that calls the Scientific Subroutines under the RT-11 operating system.

However, the appendix provides only general information about creating programs. It does not include detailed information about the FORTRAN IV programming language, the FORTRAN IV compiler, or the RT-11 operating system. For additional information on these topics, refer to the appropriate RT-11 or FORTRAN IV documentation cited in this appendix.

A.1 INSTALLATION REQUIREMENTS

Instructions in this appendix require that:

1. All files acted upon by system programs are on the default device, DK:, unless otherwise noted in this appendix.

2. You use the default file types which are:

.FOR	FORTRAN source files
.OBJ	object files and object library files
.SAV	image (executable) background files

3. The FORTRAN IV compiler, FORTRA.SAV, has already been built and resides on the system device, SY:.

4. The FORTRAN Object Time System, OTS.SAV, has been built and added to the system library, SYSLIB.OBJ, which resides on the system device, SY:.

5. The system utility programs PIP.SAV, DUP.SAV, the RT-11 linker, LINK.SAV, and the RT-11 librarian, LIBR.SAV, have been installed and reside on the system device, SY:.

6. All necessary system programs and files are installed and reside on the system device, SY:.

A.2 INSTALLING THE SCIENTIFIC SUBROUTINES SOFTWARE

Installing the Scientific Subroutines software consists of copying the SSP distribution volumes and making any necessary corrections to the SSP software.

A.2.1 Copying the Distribution Volumes

Because all storage media can be adversely affected by environmental conditions, vandalism, and human error, you should keep several copies of any software that cannot be easily re-created. Copy the SSP distribution volumes and store them in a safe place. Use the distribution volumes only when copying the SSP software. Use only copies for all other procedures described in this appendix.

The distribution kit containing the SSP software has up to two volumes, each with a file structure that RT-11 can read. If your kit has two volumes, copy each of them. The procedure for copying the distribution volumes varies depending on how many mass storage devices you have. If you have three or more mass storage devices, follow the procedures in Section A.2.1.1. If you have only two mass storage devices, follow the procedure in Section A.2.1.2.

NOTE

Instructions for copying one volume to another use the SQUEEZE command rather than the COPY command. The SQUEEZE command copies the protection code of files; the COPY command does not. See Section A.2.2 for information about how to remove and restore the protection code of files.

A.2.1.1 Copying with Three or More Mass Storage Devices - To copy the SSP distribution volumes with three or more storage devices, do the following:

1. Load a distribution volume.
2. Load a blank storage volume.
3. Format the blank storage volume if necessary. If you use RK05 disks, format each disk. If you use RX02 drives, format each diskette to be either single density or double density. Other storage media cannot be formatted.

   Use the FORMAT command to format your disks and diskettes. The following example formats an RK05 disk:

```
.FORMAT RK1:(RET)
RK1:/FORMAT-Are you sure?Y(RET)
?FORMAT-I-Formatting complete
```

When you use the FORMAT command, diskettes on RX02 drives are formatted to be double density by default. If you want them to be single density, use the /SINGLEDENSITY switch with the FORMAT command. The following example formats a diskette to be single density:

```
.FORMAT/SINGLEDENSITY DY1:(RET)
DY1:/FORMAT-Are you sure?Y(RET)
?FORMAT-I-Formatting complete
```

See the RT-11 System User's Guide for more information about the FORMAT command.

4. Initialize the blank storage volume. If both your distribution and storage volumes are diskettes, the storage volume requires five directory segments in order to copy the SSP distribution volumes. Since the default number of directory segments on a diskette is four, use the /SEGMENTS switch with the INITIALIZE command to allocate five directory segments. In addition, use the /BADBLOCKS switch to search for and isolate bad blocks. Type:

```
.INITIALIZE/BADBLOCKS/SEGMENTS:5 dvn:(RET)
dvn:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected dvn:
```

If you have any other kind of distribution volume, or if your storage volumes are not diskettes, you can use the default number of directory segments. Use the INITIALIZE command with the /BADBLOCKS switch and omit the /SEGMENTS switch.

See the RT-11 System User's Guide for more information about the INITIALIZE command and what the system does if it finds bad blocks.

5. Use the SQUEEZE command with the /OUTPUT switch to copy files from the distribution volume to the blank storage volume. Using the SQUEEZE command with the /OUTPUT switch copies all files from the input (distribution) volume to the beginning of the output (storage) volume and consolidates all empty space on the output volume at the end of that volume. The following example copies files from dv1: to dv2:

```
.SQUEEZE/OUTPUT:dv2: dv1:(RET)
```

See the RT-11 System User's Guide for more information about the SQUEEZE command and copying files.

A.2.1.2 **Copying with Only Two Mass Storage Devices** - If you have only two mass storage devices, you cannot copy each SSP distribution volume directly since the RT-11 system volume occupies one of your mass storage devices. To copy the distribution volumes under these circumstances, do the following:

1. Load a blank storage volume.

2. Format the blank storage volume if necessary. If you use RK05 disks, format each disk. If you use RX02 drives, format each diskette to be either single density or double density. Other storage media cannot be formatted.

Use the FORMAT command to format your disks and diskettes. The following example formats an RK05 disk:

```
.FORMAT RK1:(RET)
RK1:/FORMAT-Are you sure? Y(RET)
?FORMAT-I-Formatting complete
```

When you use the FORMAT command, diskettes on RX02 drives are formatted to be double density by default. If you want them to be single density, use the /SINGLEDENSITY switch with the FORMAT command. The following example formats a diskette to be single density:

```
.FORMAT/SINGLEDENSITY DY1:(RET)
DY1:/FORMAT-Are you sure? Y(RET)
?FORMAT-I-Formatting complete
```

See the RT-11 System User's Guide for more information about the FORMAT command.

3. Initialize the blank storage volume. If both your distribution and storage volumes are diskettes, the storage volume requires five directory segments in order to copy the SSP distribution volume. Since the default number of directory segments on a diskette is four, use the /SEGMENTS switch with the INITIALIZE command to allocate five directory segments. In addition, use the /BADBLOCKS switch to search for and isolate bad blocks. Type:

```
.INITIALIZE/BADBLOCKS/SEGMENTS:5 dvn:(RET)
dvn:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected dvn:
```

If you have any other kind of distribution volume, or if your storage volumes are not diskettes, you can use the default number of directory segments. Use the INITIALIZE command with the /BADBLOCKS switch and omit the /SEGMENTS switch.

See the RT-11 System User's Guide for more information about the INITIALIZE command and what the system does if it finds bad blocks.

4. Use the SQUEEZE command with both the /WAIT and the /OUTPUT switches to copy files from the distribution volume to the blank storage volume. Using the /WAIT switch causes RT-11 to pause before copying files, allowing you to load a blank storage volume, if necessary, and to remove the RT-11 system volume from the system device. Then you can load an SSP distribution volume in the system device. RT-11 prompts you with messages telling you when to do each of these things. Answer Y (RET) to each message when you are ready to continue.

The /OUTPUT switch causes RT-11 to copy files from the input (distribution) volume to the beginning of the output (storage) volume and consolidates all empty space on the output volume at the end of that volume. After RT-11 finishes copying the files, it prompts you with a message telling you to replace the system volume in the system device. The following example copies files from dv0: to dv1:

```
.SQUEEZE/WAIT/OUTPUT:dv1: dv0:(RET)
Mount output volume in dv1:; Continue? Y(RET)
Mount input volume in dv0:; Continue? Y(RET)
```

At this point RT-11 copies the files and then types the following message on your terminal:

```
Mount system volume in dv0:; Continue? Y (RET)
```

See the RT-11 System User's Guide for more information about the SQUEEZE command and copying files.

A.2.2 File Protection

Remember that all files in the SSP distribution volumes have been protected. Never remove this protection. However, after copying the distribution volumes, you may remove the protection of files from your copies, if you wish, by using the RENAME command with the /NOPROTECTION switch. To protect an unprotected file, use the /PROTECTION switch with the RENAME command.

See the RT-11 System User's Guide for more information about the RENAME command.

A.2.3 Making Corrections

From time to time, the RT-11 Software Dispatch publishes corrections that you must make to the SSP software. The dispatch also publishes instructions on how to make the corrections.

You only need to make corrections published in the dispatch for version 1.3 of the SSP software. Corrections that were published in the dispatch for previous versions of SSP have already been included in version 1.3.

Never make corrections to your distribution volumes. Make corrections to your copies. After making any corrections, copy your software again.

A.3 VERIFYING THE SCIENTIFIC SUBROUTINES SOFTWARE

The SSP distribution volumes contain indirect-command files that you use to verify that you performed the installation procedure correctly, and that your software is in good working order. Section A.3.1 describes the indirect-command files. Section A.3.2 describes the procedure itself and tells you how to execute it. Section A.3.3 describes conditions that can cause errors.

A.3.1 The Indirect-Command Files

Each indirect-command file calls other files:

1. A FORTRAN test program, which is a source file and has the file type .FOR.

2. Data files, which some of the FORTRAN programs use as input data to the SSP subroutines. Data files have the type .DAT.

When you run an indirect-command file, it does the following:

1. Directs the FORTRAN IV compiler to compile a FORTRAN IV test program and one or more of the Scientific Subroutines. The FORTRAN program calls up to sixteen of the Scientific Subroutines, thus testing if they work correctly.

   Compiling creates one or more object files called TMPSSP.n where n is a number less than ten. TMPSSP.n is a "temporary" file because it resides temporarily on a storage device that you assign to receive output files. (Section A.3.2 explains how to assign devices.)

2. Instructs the RT-11 linker to link the object file TMPSSP.n. The linker links the file and creates an executable file called TMPSSP.n.

3. Directs the test program TMPSSP.n to run. Some test programs require a file of input data. Input-data files are included in the distributed SSP software. The test program uses the input-data file automatically if it needs one. If your software is sound, the test program runs and types some data on your terminal. This data results from calculations performed by the test programs and the subroutines.

4. Deletes all files called TMPSSP (regardless of file type). Before running an indirect-command file, make sure you have no files called TMPSSP on the device you assign to receive temporary output files.

Make sure that you have enough contiguous space on your storage device to receive the temporary output files TMPSSP.n. Table A-1 shows how much contiguous space you need. If you think you do not have enough contiguous space, use the SQUEEZE command to move all the files on your storage volume to the beginning of the volume. See the RT-11 System User's Guide for information about the SQUEEZE command.

NOTE

If you squeeze a system volume, the system will automatically reboot after moving the files to the beginning of the volume. If you need to perform a squeeze, do so before you begin the test procedure that follows.

Table A-1 lists the indirect-command files as well as the following:

- The Scientific Subroutines that each indirect-command file tests.
- Additional files required by the indirect-command file. (Additional files are the FORTRAN test program and the input-data file if one is required.)
- The amount of contiguous space required for temporary storage of output files.

Table A-1
The Indirect-Command Files

Indirect Command Files	Subroutines Tested		Test and Data Files Used	Approximate Contiguous Space Required
ANOVA.COM	AVCAL.FOR AVDAT.FOR MEANQ.FOR		ANOVA.FOR ANOVA.DAT	92 Blocks
COLROW.COM	CINT.FOR MCPY.FOR LDC.FOR MTRA.FOR	RINT.FOR SCMA.FOR SRMA.FOR	COLROW.FOR	82 Blocks
DASCR.COM	LOC.FOR SUBST.FOR TAB1.FOR		DASCR.FOR DASCR.DAT	108 Blocks
EXPON.COM	EXSMO.FOR		EXPON.FOR EXPON.DAT	57 Blocks
FACTO.COM	CORRE.FOR EIGEN.FOR LOAD.FOR	TRACE.FOR VARMX.FOR	FACTO.FOR FACTO.DAT	167 Blocks
FORR.COM	FORIF.FOR FORIT.FOR		FORR.FOR	61 Blocks
INTEG.COM	QATR.FOR RK1.FOR		INTEG.FOR	69 Blocks
MACHK1.COM	ARRAY.FOR GMADD.FOR GMPRD.FOR GMSUB.FOR	GMTRA.FOR GTPRD.FOR LOC.FOR MADD.FOR	MACHK1.FOR	94 Blocks
MACHK2.COM	CCUT.FOR CTIE.FOR DCLA.FOR DCPY.FOR LOC.FOR MATA.FOR	MCPY.FOR RCUT.FOR RTIE.FOR TPRD.FOR XCPY.FOR	MACHK2.FOR	120 Blocks
MACHK3.COM	CCPY.FOR CSUM.FOR LOC.FOR MFUN.FOR	MSUB.FOR RSRT.FOR RSUM.FOR SMPX.FOR	MACHK3.FOR	105 Blocks
MACHK4.COM	CADD.FOR CCPY.FOR CSRT.FOR CTAB.FOR LOC.FOR MFUN.FOR MPRD.FOR MSTR.FOR	RADD.FOR RCPY.FOR RECP.FOR RTAB.FOR SADD.FOR SCLA.FOR SDIV.FOR SSUB.FOR	MACHK4.FOR	158 Blocks

Table A-1 (Cont.)
The Indirect-Command Files

Indirect Command Files	Subroutines Tested		Test and Data Files Used	Approximate Contiguous Space Required
MCANO.COM	CANOR.FOR CORRE.FOR EIGEN.FOR MINV.FOR NROOT.FOR		MCANO.FOR MCANO.DAT	176 Blocks
MDISC.COM	DISCR.FOR DMATX.FOR MINV.FOR		MDISC.FOR MDISC.DAT	108 Blocks
NONLIN.COM	RTMI.FOR RTNI.FOR RTWI.FOR		NONLIN.FOR	78 Blocks
NONPAR.COM	CHISQ.FOR GAUSS.FOR KRANK.FOR RANK.FOR	SRANK.FOR TIE.FOR TWOAV.FOR UTEST.FOR	NONPAR.FOR NONPAR.DAT	132 Blocks
NPAR2.COM	MOMEN.FOR QTEST.FOR RANK.FOR TIE.FOR WTEST.FOR		NPAR2.FOR	89 Blocks
POLRG.COM	GDATA.FOR MINV.FOR MULTR.FOR ORDER.FOR		POLRG.FOR POLRG.DAT	140 Blocks
POLY1.COM	PADDM.FOR PDER.FOR PDIV.FOR PINT.FOR	PMPY.FOR PNORM.FOR PVAL.FOR	POLY1.FOR	84 Blocks
POLY2.COM	PADD.FOR PADDM.FOR PCLA.FOR PCLD.FOR PDIV.FOR PGCD.FOR	PILD.FOR PMPY.FOR PNORM.FOR PQSD.FOR PSUB.FOR PVSUB.FOR	POLY2.FOR	118 Blocks
QDINT.COM	QSF.FOR		QDINT.FOR QDINT.DAT	55 Blocks
RKGSTT.COM	RKGS.FOR		RKGSTT.FOR	71 Blocks
RK2INT.COM	RK2.FOR		RK2INT.FOR RK2INT.DAT	48 Blocks
*SMPRT.COM	POLRT.FOR		SMPRT.FOR SMPRT.DAT	61 Blocks

Table A-1 (Cont.)
The Indirect-Command Files

Indirect Command Files	Subroutines Tested	Test and Data Files Used	Approximate Contiguous Space Required
SOLVEN.COM	GMPRD.FOR MINV.FOR SIMQ.FOR	SOLVEN.FOR	79 Blocks
STAT.COM	ABSNT.FOR TAB2.FOR BOUND.FOR TALLY.FOR GAUSS.FOR TTSTT.FOR SUBMX.FOR	STAT.FOR	138 Blocks
TABLE1.COM	CS.FOR GAMMA.FOR LEP.FOR	TABLE1.FOR	63 Blocks
TABLE2.COM	BESI.FOR BESJ.FOR BESK.FOR BESY.FOR	TABLE2.FOR	103 Blocks
TABLE3.COM	CEL1.FOR CEL2.FOR EXPI.FOR SICI.FOR	TABLE3.FOR	82 Blocks
TIMSER.COM	AUTO.FOR CROSS.FOR GAUSS.FOR SMO.FOR	TIMSER.FOR	69 Blocks

*When you run the indirect-command file SMPRT.COM, you may get an overflow error message from the subroutine POLRT.FOR (see Section 3.8.1). However, POLRT.FOR can deal with the overflow error. You can ignore the error message.

A.3.2 The Verification Procedure

SSP contains more than 100 subroutines. Verify only those you plan to use. To use the indirect-command files for verifying subroutines, do the following:

1. Make sure that the requirements listed in Section A.1 are true for your system.

2. Perform a squeeze if you need to (see Section A.3.1).

3. Assign logical device names to:

   - the device containing all the files you will use for input. Assign this device the logical name "IN" for "input device." Type:

     .ASSIGN dvn: IN: RET

   - the device used for storage of the temporary output files TMPSSP.n created by the indirect-command file when it runs. Assign this device the logical name "TM" for "temporary storage device." Type:

     .ASSIGN dvn: TM: RET

4. Consult table A-1. Find the name of a subroutine you want to test and the indirect command file that verifies it.

5. Run the indirect-command file. Type:

   .@IN:file RET

   where: @ is the symbol that indicates you want to run an indirect-command file.

   file is the name of the indirect-command file you want to run.

   This command runs the indirect-command file you specified, which in turn runs the FORTRAN test program. If your software is sound and if you have performed all the above instructions correctly, the FORTRAN test program runs to completion and types some data on your terminal followed by the message

   STOP--file verification procedure successful

   where: file is the name of the indirect-command file you ran.

A.3.3 Error Conditions

If an indirect-command file finds an error, it stops running and types an error message on your terminal. However, this does not necessarily mean that your software is deficient or that it cannot be verified. An indirect-command file can stop running because some requirement has not been met even though your software is sound.

Some typical conditions that can cause error messages follow. If one of the indirect-command files will not run, and you get an error message, check to see that you have not neglected one of these conditions.

- A necessary system program such as the FORTRAN IV compiler or a utility program such as the RT-11 linker or PIP or DUP is not on your system device.

- You assigned the input device incorrectly. That is, the device you assigned to be the input device does not contain the volume where the Scientific Subroutines files that you will use as input reside.

- You assigned the output device incorrectly. That is, the device you assigned to be the output device does not have enough contiguous storage space.

- Your system device or your input or output device was write-protected when you tried to run an indirect-command file.

It is impossible to anticipate all conditions that can cause errors that are not the fault of the software. If you receive an error message, first check to see that all the requirements listed in Section A.1 are true for your system. Then, if you need to, see the RT-11 System Message Manual to help you determine the cause of the error and correct it.

Finally, if one of the indirect-command files will not run, even though you have not made any errors of the kind described above or neglected any of the requirements listed in Section A.1, you may have received a defective copy of your Scientific Subroutines software. Contact DIGITAL for more information.

A.4 STORING THE SCIENTIFIC SUBROUTINES

After determining that the Scientific Subroutines you want to use are sound, copy them to your system volume or to the development volume where you store your FORTRAN programs.

A.5 CREATING A PROGRAM THAT CALLS THE SCIENTIFIC SUBROUTINES

To create a FORTRAN IV program that calls the Scientific Subroutines, do the following:

1. Write and check your program.

2. Use one of the RT-11 editors, such as EDIT, to enter your program into a source file. The EDIT command with the /CREATE switch invokes EDIT and tells it to create a new file. Type:

   .EDIT/CREATE prog.FOR (RET)

   where: prog is the name of your FORTRAN source program.

   For information about entering the text of your file, making changes, and displaying the file, see the Introduction to RT-11 and the RT-11 System User's Guide.

NOTE

Always specify a file type when you use an RT-11 editor. RT-11 editors do not use default file types. In this case, give your source file the type .FOR since that is the default file type the FORTRAN IV compiler uses when it compiles your program.

3. Use the FORTRAN IV compiler to create object files of your program and of the Scientific Subroutines you wish to use. Type:

```
.FORTRAN prog,subl,sub2,...subn RET
```

where: prog is the name of your FORTRAN source program.

subl is the name of the first Scientific Subroutine you wish to compile.

subn is the name of the last Scientific Subroutine you wish to compile.

In the FORTRAN command, you can specify up to six files per line. If you need to compile more than six files at one time, see the RT-11 System User's Guide for more information.

NOTE

When you compile some of the Scientific Subroutines, the RT-11 FORTRAN IV compiler prints a warning message. You can ignore this message. See Appendix C to find which subroutines cause the compiler to print a warning message.

Frequently used Scientific Subroutines can be compiled in advance. Then you only need to compile your main program.

4. Use the RT-11 linker to link the object file of your program with the object files of your Scientific Subroutines. Type:

```
.LINK prog,subl,sub2,...subn RET
```

where: prog is the name of your program.

subl is the name of the first Scientific Subroutines object file you want to link.

subn is the name of the last Scientific Subroutines object file you want to link.

To avoid linking individual subroutines to your program, you can place the subroutines in a library. See Section A.6.

5. Run the program. Type:

```
.RUN prog RET
```

where: prog is the name of your executable program.

For more information about compiling, linking, and running FORTRAN IV programs, see the RT-11 System User's Guide, and the RT-11/RSTS/E FORTRAN IV User's Guide.

A.6 USING LIBRARIES

Once you decide which of the Scientific Subroutines you will use most frequently, you can link them more easily to your programs by placing them in a library. When you place them in a library, you do not need to list each individual subroutine in the link command line. You only need to list the library name. When you place them in the system library, SYSLIB.OBJ, you do not even need to list the library name in the link command line. The RT-11 linker automatically searches the system library for a subroutine or function needed by a program.

For example, suppose your compiled program, MYPROG.OBJ, calls the compiled subroutines, ABSNT.OBJ, BOUND.OBJ, and GAUSS.OBJ. To link MYPROG, you have to type:

```
.LINK MYPROG,ABSNT,BOUND,GAUSS RET
```

If you place the subroutines in a library called, for example, MYLIB.OBJ, you link MYPROG by typing:

```
.LINK MYPROG,MYLIB RET
```

If you add the subroutines to the system library, SYSLIB.OBJ, you link MYPROG by typing:

```
.LINK MYPROG RET
```

The RT-11 librarian program LIBR.SAV lets you add object files or object file libraries to SYSLIB. When you do so, however, you must remove certain global symbols from SYSLIB's directory for the library to function properly. Remove the global symbol $OVRH each time you add files or libraries to SYSLIB. Also remove $ERRS and $ERRTB if you previously added the FORTRAN IV OTS library to SYSLIB.

To add individual object files or a library to a system library containing the distributed SYSLIB.OBJ file and the FORTRAN IV OTS library, use the LIBRARY command with the /REMOVE switch. The LIBRARY command invokes LIBR.SAV. The /REMOVE switch allows you to remove a global symbol from a library directory. The following example adds an object file residing on dv1: to SYSLIB on dv0 and removes the appropriate global symbols:

```
.LIBRARY/REMOVE dv0:SYSLIB dv1:file.OBJ RET
Global? $OVRH RET
Global? $ERRS RET
Global? $ERRTB RET
Global? (any other global previously removed
         from either library) RET
Global? RET
```

where: file: is the name of the file or library you want to place in the library.

If you fail to remove any of the necessary global symbols when you execute this command, you will get a warning message for each symbol you failed to remove. An example of the warning message is:

```
?LIBR-W-Illegal insert of $OVRH
```

If you forget which global symbols to remove, create a dummy library file by typing a command such as the following:

```
.LIBRARY/CREATE dummy SYSLIB
```

(RET)

where: dummy is the name of a dummy library file.

Executing this command will produce error messages that list all the global symbols you should remove. After noting the names of the global symbols, delete the dummy library file and type the correct command for adding files to SYSLIB.

For more information about the LIBRARY command and the RT-11 librarian program, LIBR.SAV, see the RT-11 System User's Guide.

APPENDIX B

INSTALLING, VERIFYING, AND USING SSP UNDER RSX-11M/M-PLUS

This appendix explains how to install and verify the Scientific Subroutines Package (SSP) software and how to create a FORTRAN program that calls the Scientific Subroutines under the RSX-11M/M-PLUS operating systems.

However, the appendix provides only general information about creating programs. It does not include detailed information about the FORTRAN IV or FORTRAN 77 programming languages, the FORTRAN IV or FORTRAN 77 compilers, or the RSX-11M/M-PLUS operating systems. For additional information on these topics, refer to the appropriate RSX-11M/M-PLUS, FORTRAN IV, or FORTRAN 77 documentation cited in this appendix.

B.1 INSTALLATION REQUIREMENTS

Instructions in this appendix require that:

1. All files acted upon by system programs are on the default device, SY: under your UIC unless otherwise noted in this appendix.

2. You use the default file extensions which are:

   .FTN FORTRAN source files
   .OBJ object files
   .OLB object library files
   .TSK task-image (executable) files

3. Either the FORTRAN IV or the FORTRAN 77 compiler has already been built and installed, and resides on the system device, SY:.

4. Either the FORTRAN Object Time System (FOROTS) or the FORTRAN 77 Object Time System (F4POTS) has been built and added to the system library SYSLIB.OLB which resides on the system device, SY:. Make sure that only the object time system you will use is added to the system library.

5. The system utility programs PIP, DSC, FLX, and the RSX-11M/M-PLUS task builder, TKB, have been installed and reside on the system device, SY:.

6. All necessary system programs and files are installed and reside on the system device, SY:.

7. All tasks and MCR functions are installed.

8. The system is set to accept MCR rather than DCL commands.

B.2 INSTALLING THE SCIENTIFIC SUBROUTINES SOFTWARE

Installing the Scientific Subroutines software consists of copying the SSP distribution volume and making any necessary corrections to the SSP software.

B.2.1 Copying the Distribution Volume

Because all storage media can be adversely affected by environmental conditions, vandalism, and human error, you should keep several copies of any software that cannot be easily re-created. Copy the SSP distribution volume and store it in a safe place. Use the distribution volume only when copying the SSP software. Use only copies for all other procedures described in this appendix.

The SSP software is distributed on a single volume. Most of the Scientific Subroutines distribution volumes for RSX-11M/M-PLUS are in FILES-11 format which RSX-11M/M-PLUS can read. However, magnetic tape distribution kits are in DOS-11 format which RSX-11M/M-PLUS cannot read.

Copy procedures vary depending on the type of distribution volume and the number of mass storage devices you have:

- If your distribution volume is a FILES-11 volume such as an RK05, RK06, RL01, or RL02 disk and you have three or more mass storage devices, follow the instructions in Section B.2.1.1.
- If your distribution volume is a FILES-11 volume and you have only two mass storage devices, follow the instructions in Section B.2.1.2.
- If your distribution volume is a DOS-11 formatted magnetic tape, follow the instructions in Section B.2.1.3.

B.2.1.1 Copying a FILES-11 Distribution Volume with Three or More Mass Storage Devices - To copy a FILES-11 distribution volume, do the following:

1. Load the distribution volume. Allocate the drive and mount the distribution volume. The volume label for the Scientific Subroutines is SSP. The User Identification Code (UIC) is [200,200]. Type:

   ```
   >ALL dvn:(RET)
   >MOU dvn:SSP(RET)
   ```

2. Load a blank storage volume. Allocate the drive. Type:

   ```
   >ALL dvn:(RET)
   ```

3. If you use RSX-11M-PLUS, mount the blank storage volume foreign since it has not yet been initialized as a FILES-11 volume. Type:

   ```
   >MOU dvn:/FOR(RET)
   ```

4. Format the blank storage volume if necessary. If you use RK05 disks, format each disk. If you use RX02 drives, format each diskette to be either low (single) or high (double) density. Other storage media can not be formatted.

   Use the RSX utility FMT to format your disks and diskettes. The following example formats an RK05 disk:

```
>FMT DK2:(RET)

** WARNING - DATA WILL BE LOST ON DK2:  **

CONTINUE?  [Y OR N] Y(RET)

START FORMATTING

OPERATION COMPLETE

>
```

   When you use FMT, diskettes on RX02 drives are formatted to be low (single) density by default. If you want them to be high (double) density, use the /DENS=HIGH option. The following example formats an RX02 diskette to be double density:

```
>FMT DY1:/DENS=HIGH(RET)

** WARNING - DATA WILL BE LOST ON DY1:  **

CONTINUE?  [Y OR N] Y(RET)

START FORMATTING

OPERATION COMPLETE

>
```

   See the RSX-11M/M-PLUS Utilities Manual for more information about FMT.

5. Run the Bad Block Locator Utility (BAD) to search for bad blocks on the storage volume. The /LI switch causes BAD to list at your terminal the decimal number of any bad blocks found. Type:

```
>BAD dvn:/LI(RET)
```

   If BAD finds bad blocks, it types a message for each one found. An example of the message is:

```
BAD -- dvn:  BAD BLOCK FOUND - LBN = nnnnnn
```

   LBN means Logical Block Number, and nnnnnn is the decimal number of the bad block found. When BAD finishes reading the storage volume it types a message telling how many bad blocks it found:

```
BAD -- dvn:  TOTAL BAD BLOCKS= n
```

   If BAD found none, it types 0. If your volume has bad blocks, isolate them using the /BAD=[AUTO] option with the INI command (see step 6).

If BAD types any other message, see Chapter 6 of the RSX-11M/M-PLUS Utilities Manual to find out what the message means and what to do about it.

6. Initialize the blank storage volume so it will be in FILES-11 format. Use the MCR command INI with the /BAD=[AUTO] option to isolate bad blocks on the volume. Type:

```
>INI dvn:/BAD=[AUTO]<RET>

INI -- NO BAD BLOCK DATA FOUND
>
```

If you use RSX-11M-PLUS, INI does not type the above message; it just displays its prompt to indicate it found no bad blocks.

See the RSX-11M/M-PLUS MCR Operations Manual for more information about using the MCR command INI.

7. If you use RSX-11M-PLUS, your storage volume is still mounted foreign although it is now in FILES-11 format. Dismount it. Type:

```
>DMOU dvn:<RET>
```

8. For both RSX-11M and RSX-11M-PLUS, mount the volume as a FILES-11 volume. Type:

```
>MOU dvn:<RET>
```

9. Use the RSX Disc Save and Compress Utility (DSC) to copy files from the distribution volume to the storage volume. The following example copies files from a distribution volume on drive 1 to a storage volume on drive 2:

```
>DSC dv2:=dv1:<RET>
>
```

See the RSX-11M/M-PLUS Utilities Manual for more information about using DSC to copy files.

B.2.1.2 **Copying a FILES-11 Distribution Volume with only Two Mass Storage Devices** - If you have only two mass storage devices, you cannot copy the SSP distribution volume directly since the RSX-11M/M-PLUS system volume occupies one of your mass storage devices.

To copy the distribution volume under these circumstances, use the stand-alone version of the RSX utility program DSC, named DSCSYS. Once booted, DSCSYS no longer needs the system volume. You can unload the system volume thus freeing the system device. Then you can load the SSP distribution volume in the system device and begin copying files. The following procedure explains how to copy the distribution volume using DSCSYS.

NOTE

Since this procedure requires you to remove the system volume and halt the

processor, make sure that no one else is using the system when you execute the procedure.

1. Load a blank storage volume. Allocate the drive it is on. Type:

   >ALL dvn:(RET)

2. If you use RSX-11M-PLUS, mount the blank storage volume foreign since it has not yet been initialized as a FILES-11 volume. Type:

   >MOU dvn:/FOR(RET)

3. Format the blank storage volume if necessary. If you use RK05 disks, format each disk. If you use RX02 drives, format each diskette to be either low (single) or high (double) density. Other storage media can not be formatted.

   Use the RSX utility FMT to format your disks and diskettes. The following example formats a RK05 disk:

   ```
   >FMT DK1:(RET)

   ** WARNING - DATA WILL BE LOST ON DK1:  **

   CONTINUE?  [Y OR N] Y(RET)

   START FORMATTING

   OPERATION COMPLETE

   >
   ```

   When you use FMT, diskettes on RX02 drives are formatted to be low (single) density by default. If you want them to be high (double) density, use the /DENS=HIGH option. The following example formats an RX02 diskette to be double density:

   ```
   >FMT DY1:/DENS=HIGH(RET)

   ** WARNING - DATA WILL BE LOST ON DY1:  **

   CONTINUE?  [Y OR N] Y(RET)

   START FORMATTING

   OPERATION COMPLETE

   >
   ```

   See the RSX-11M/M-PLUS Utilities Manual for more information about FMT.

4. Run the Bad Block Locator Utility (BAD) to search for bad blocks on the storage volume. The /LI switch causes BAD to list at your terminal the decimal number of any bad blocks found. Type:

   ```
   >BAD dvn:/LI(RET)

   BAD -- NO BAD BLOCK DATA FOUND
   ```

If BAD finds bad blocks, it types a message for each one found. An example of the message is:

```
BAD -- dvn:  BAD BLOCK FOUND - LBN = nnnnnn
```

LBN means Logical Block Number, and nnnnnn is the decimal number of the bad block found. When BAD finishes reading the storage volume, it types a message telling how many bad blocks it found:

```
BAD -- dvn:  TOTAL BAD BLOCKS= n
```

If BAD found none, it types 0. If your volume has bad blocks, isolate them using the /BAD=[AUTO] option with the INI command (see step 5).

If BAD types any other message, see Chapter 6 of the RSX-11M/M-PLUS Utilities Manual to find out what the message means and what to do about it.

5. Initialize the blank storage volume so it will be in FILES-11 format. Use the MCR command INI with the /BAD=[AUTO] option to isolate bad blocks on the volume. Type:

```
>INI dvn:/BAD=[AUTO]RET
```

```
INI -- NO BAD BLOCK DATA FOUND
>
```

If you use RSX-11M-PLUS, INI does not type the above message; it just displays its prompt to indicate it found no bad blocks.

See the RSX-11M/M-PLUS MCR Operations Manual for more information about using the MCR command INI.

6. Boot DSCSYS. The UIC is [1,51]. Type:

```
>BOOT [1,51]DSCSYSRET
RSX11S V4.0 BL32 DISK SAVE AND COMPRESS UTILITY V 4.0
```

DSCSYS displays its prompt. It is now booted and ready for input.

```
DSC>
```

7. Remove the RSX-11M/M-PLUS system volume and load a copy of the SSP distribution volume in the system device. Now instruct DSCSYS to begin copying files. The following example copies files from a distribution volume, on drive 0 to a storage volume on drive 1:

```
DSC>dv1:=dv0:RET
```

There will be warning messages that the disks are not bootable, then the copying begins. When it completes, DSCSYS displays its prompt.

```
DSC>
```

8. Now halt the processor. Remove the SSP distribution volume from the system device and replace the system volume. Boot RSX-11M or RSX-11M-PLUS.

See the RSX-11M/M-PLUS Utilities Manual for more information about DSC.

B.2.1.3 Copying a DOS-11 Distribution Volume - To copy a DOS-11 distribution magnetic tape, do the following:

1. Load the distribution volume. Allocate the drive and mount the distribution volume. The User Identification Code (UIC) is [200,200]. Type:

   >ALL dvn:(RET)
   >MOU dvn:(RET)

2. Load a blank storage volume. Allocate the drive it is on. Type:

   >ALL dvn:(RET)

3. If you use RSX-11M-PLUS, mount the blank storage volume foreign since it has not yet been initialized as a FILES-11 volume. Type:

   >MOU dvn:/FOR(RET)

4. Format the blank storage volume if necessary. If you use RK05 disks, format each disk. If you use RK02 drives, format each diskette to be either low (single) or high (double) density. Other storage media can not be formatted.

   Use the RSX utility FMT to format your disks and diskettes. The following example formats an RK05 disk:

   >FMT DK1:(RET)

   ** WARNING - DATA WILL BE LOST ON DK1: **

   CONTINUE? [Y or N] Y(RET)

   START FORMATTING

   OPERATION COMPLETE

   >

   When you use FMT, diskettes on RX02 drives are formatted to be low (single) density by default. If you want them to be high (double) density, use the /DENS=HIGH option. The following example formats an RX02 diskette to be double density:

   >FMT DY1:/DENS=HIGH(RET)

   ** WARNING - DATA WILL BE LOST ON DY1: **

   CONTINUE? [Y OR N] Y(RET)

   START FORMATTING

   OPERATION COMPLETE

   >

See the RSX-11M/M-PLUS Utilities Manual for more information about FMT.

5. Run the Bad Block Locator Utility (BAD) to search for bad blocks on the storage volume. The /LI switch causes BAD to list at your terminal the decimal number of any bad blocks found. Type:

   >BAD dvn:/LI(RET)

   BAD -- NO BAD BLOCK DATA FOUND

   If BAD finds bad blocks, it types a message for each one found. An example of the message is:

   BAD -- dvn: BAD BLOCK FOUND - LBN = nnnnnn

   LBN means Logical Block Number, and nnnnnn is the decimal number of the bad block found. When BAD finishes reading the storage volume, it types a message telling how many bad blocks it found:

   BAD -- dvn: TOTAL BAD BLOCKS= n

   If BAD found none, it types 0. If your volume has bad blocks, isolate them using the /BAD=[AUTO] option with the INI command (see step 6).

   If BAD types any other message, see Chapter 6 of the RSX-11M/M-PLUS Utilities Manual to find out what the message means and what to do about it.

6. Initialize the blank storage volume so it will be in FILES-11 format. Use the MCR command INI with the /BAD=[AUTO] option to isolate bad blocks on the volume. Type:

   >INI dvn:/BAD=[AUTO](RET)

   INI -- NO BAD BLOCK DATA FOUND
   >

   If you use RSX-11M-PLUS, INI does not type the above message; it just displays its prompt to indicate it found no bad blocks.

   See the RSX-11M/M-PLUS MCR Operations Manual for more information about using the MCR command INI.

7. If you use RSX-11M-PLUS, your storage volume is still mounted foreign although it is now in FILES-11 format. Dismount it. Type:

   >DMOU dvn:(RET)

8. For both RSX-11M and RSX-11M-PLUS, mount the volume as a FILES-11 volume. Type:

   >MOU dvn:(RET)

9. Create a UIC of [200,200] on the storage volume. Use the MCR command UFD (User File Directory). Type:

   >UFD dvn:[200,200](RET)

10. Use the RSX utility FLX to copy files from the DOS-11 distribution tape to your FILES-11 storage volume. The /RS switch identifies the storage volume as a FILES-11 volume, and the /DO switch identifies the distribution volume as a DOS-11 volume. The wildcard construction, *.*, tells RSX-11M/M-PLUS that you wish to copy the most recent version of all files. (You cannot specify version numbers on a DOS-11 formatted volume.)

    The following example copies files from a DOS-11 distribution volume on drive 0 to a FILES-11 storage volume on drive 1:

    ```
    >FLX dv1:[200,200]/RS=dv0:[200,200]*.*/DO RET
    >
    ```

    See the RSX-11M/M-PLUS Utilities Manual for more information about FLX.

B.2.2 Making Corrections

From time to time, the RSX-11M/S-RSX-11M-PLUS Software Dispatch publishes corrections you must make to the SSP software. The dispatch also publishes instructions on how to make the corrections.

You only need to make corrections published in the dispatch for version 1.3 of the SSP software. Corrections that were published in the dispatch for previous versions of SSP have already been included in version 1.3.

Never make corrections to your distribution volume. Make corrections to your copies. After making any corrections, copy your software again.

B.3 VERIFYING THE SCIENTIFIC SUBROUTINES SOFTWARE

The SSP distribution volume contains indirect-command files that you use to verify that you performed the installation procedure correctly, and that your software is in good working order. Section B.3.1 describes the indirect-command files. Section B.3.2 describes the procedure itself and tells you how to execute it. Section B.3.3 describes conditions that can cause errors.

B.3.1 The Indirect-Command Files

Each indirect-command file calls other files:

1. A FORTRAN test program, which is a source file and has the extension .FTN.

2. Data files which some of the FORTRAN programs use as input data to the SSP subroutines. Data files have the type .DAT.

When you run an indirect-command file, it does the following:

1. Directs the FORTRAN IV or FORTRAN 77 compiler to compile a FORTRAN test program and one or more of the Scientific Subroutines. The FORTRAN program calls up to sixteen of the Scientific Subroutines, thus testing if they work correctly.

   Compiling creates one or more object files called TMPSSP.n where n is a number less than ten. TMPSSP.n is a "temporary" file because it resides temporarily on a storage device that you assign to receive output files. (Section B.3.2 explains how to assign devices.)

2. Instructs the RSX-11M task-builder to generate the object file TMPSSP.n. The task-builder generates the file and creates a task-image (executable) file called TMPSSP.n.

3. Directs the test program TMPSSP.n to run. Some test programs require a file of input data. Input-data files are included in the distributed SSP software. The test program uses the input-data file automatically if it needs one. If your software is sound, the test program runs and types some data on your terminal. This data results from calculations performed by the test programs and the subroutines.

4. Deletes all files called TMPSSP (regardless of file type). Before running an indirect-command file, make sure you have no files called TMPSSP on the device you assign to receive temporary output files.

Table B-1 lists the indirect-command files as well as the following:

- The Scientific Subroutines that each indirect-command file tests.
- Additional files required by the indirect-command file. (Additional files are the FORTRAN test program and the input-data file if one is required.)
- The amount of storage space required for temporary storage of output files.

Table B-1
The Indirect-Command Files

Indirect Command Files	Subroutines Tested		Test and Data Files Used	Approximate Storage Space Required
ANOVA.CMD	AVCAL.FTN AVDAT.FTN MEANQ.FTN		ANOVA.FTN ANOVA.DAT	114 Blocks
COLROW.CMD	CINT.FTN MCPY.FTN LOC.FTN MTRA.FTN	RINT.FTN SCMA.FTN SRMA.FTN	COLROW.FTN	83 Blocks
DASCR.CMD	LOC.FTN SUBST.FTN TAB1.FTN		DASCR.FTN DASCR.DAT	124 Blocks
EXPON.CMD	EXSMO.FTN		EXPON.FTN EXPON.DAT	77 Blocks
FACTO.CMD	CORRE.FTN EIGEN.FTN LOAD.FTN	TRACE.FTN VARMX.FTN	FACTO.FTN FACTO.DAT	172 Blocks
FORR.CMD	FORIF.FTN FORIT.FTN		FORR.FTN	68 Blocks
INTEG.CMD	QATR.FTN RK1.FTN		INTEG.FTN	78 Blocks
MACHK1.CMD	ARRAY.FTN GMADD.FTN GMPRD.FTN GMSUB.FTN	GMTRA.FTN GTPRD.FTN LOC.FTN MADD.FTN	MACHK1.FTN	106 Blocks
MACHK2.CMD	CCUT.FTN CTIE.FTN DCLA.FTN DCPY.FTN LOC.FTN MATA.FTN	MCPY.FTN RCUT.FTN RTIE.FTN TPRD.FTN XCPY.FTN	MACHK2.FTN	131 Blocks
MACHK3.CMD	CCPY.FTN CSUM.FTN LOC.FTN MFUN.FTN	MSUB.FTN RSRT.FTN RSUM.FTN SMPX.FTN	MACHK3.FTN	112 Blocks

Table B-1 (Cont.)
The Indirect-Command Files

Indirect Command Files	Subroutines Tested		Test and Data Files Used	Approximate Storage Space Required
MACHK4.CMD	CADD.FTN CCPY.FTN CSRT.FTN CTAB.FTN LOC.FTN MFUN.FTN MPRD.FTN MSTR.FTN	RADD.FTN RCPY.FTN RECP.FTN RTAB.FTN SADD.FTN SCLA.FTN SDIV.FTN SSUB.FTN	MACHK4.FTN	156 Blocks
MCANO.CMD	CANOR.FTN CORRE.FTN EIGEN.FTN MINV.FTN NROOT.FTN		MCANO.FTN MCANO.DAT	174 Blocks
MDISC.CMD	DISCR.FTN DMATX.FTN MINV.FTN		MDISC.FTN MDISC.DAT	119 Blocks
NONLIN.CMD	RTMI.FTN RTNI.FTN RTWI.FTN		NONLIN.FTN	84 Blocks
NONPAR.CMD	CHISQ.FTN GAUSS.FTN FRANK.FTN RANK.FTN	SRANK.FTN TIE.FTN TWOAV.FTN UTEST.FTN	NONPAR.FTN NONPAR.DAT	141 Blocks
NPAR2.CMD	MOMEN.FTN QTEST.FTN RANK.FTN TIE.FTN WTEST.FTN		NPAR2.FTN	92 Blocks
POLRG.CMD	GDATA.FTN MINV.FTN MULTR.FTN ORDER.FTN		POLRG.FTN POLRG.DAT	146 Blocks
POLY1.CMD	PADDM.FTN PDER.FTN PDIV.FTN PINT.FTN	PMPY.FTN PNORM.FTN PVAL.FTN	POLY1.FTN	86 Blocks
POLY2.CMD	PADD.FTN PADDM.FTN PCLA.FTN PCLD.FTN PDIV.FTN PGCD.FTN	PILD.FTN PMPY.FTN PNORM.FTN PQSD.FTN PSUB.FTN PVSUB.FTN	POLY2.FTN	116 Blocks
QDINT.CMD	QSF.FTN		QDINT.FTN QDINT.DAT	74 Blocks

Table B-1 (Cont.)
The Indirect-Command Files

Indirect Command Files	Subroutines Tested		Test and Data Files Used	Approximate Storage Space Required
RKGSTT.CMD	RKGS.FTN		RKGSTT.FTN	80 Blocks
RK2INT.CMD	RK2.FTN		RK2INT.FTN RK2INT.DAT	65 Blocks
*SMPRT.CMD	POLRT.FTN		SMPRT.FTN SMPRT.DAT	76 Blocks
SOLVEN.CMD	GMPRD.FTN MINV.FTN SIMQ.FTN		SOLVEN.FTN	80 Blocks
STAT.CMD	ABSNT.FTN BOUND.FTN GAUSS.FTN SUBMX.FTN	TAB2.FTN TALLY.FTN TTSTT.FTN	STAT.FTN	142 Blocks
TABLE1.CMD	CS.FTN GAMMA.FTN LEP.FTN		TABLE1.FTN	67 Blocks
TABLE2.CMD	BESI.FTN BESJ.FTN BESK.FTN BESY.FTN		TABLE2.FTN	96 Blocks
TABLE3.CMD	CEL1.FTN CEL2.FTN EXPI.FTN SICI.FTN		TABLE3.FTN	83 Blocks
TIMSER.CMD	AUTO.FTN CROSS.FTN GAUSS.FTN SMO.FTN		TIMSER.FTN	78 Blocks

*When you run any of the SMPRT indirect-command files, you may get an overflow error message from the subroutine POLRT.FTN (See Section 3.8.1). However, POLRT.FTN can deal with the overflow error. You can ignore the error message.

B.3.2 The Verification Procedure

SSP contains more than 100 subroutines. Verify only those you plan to use. To use the indirect command files for verifying subroutines, do the following:

1. Make sure that the requirements listed in Section B.1 are true for your system.

2. Assign logical device names to:

   - the device containing all the files you will use for input. Assign this device the logical name "IN" for "input device." Type:

     >ASN dvn:=IN:(RET)

   - the device used for storage of the temporary output files TMPSSP.n created by the indirect-command file when it runs. Assign this device the logical name "TM" for "temporary storage device." Type:

     >ASN dvn:=TM:(RET)

3. Consult Table B-1. Find the name of a subroutine you want to test and the indirect command file that verifies it.

4. Run the indirect-command file. After the monitor prompt appears, type:

   @IN:[200,200]file(RET)

   where: @ is the symbol that indicates you want to run an indirect-command file.

   [200,200] is the UIC for the Scientific Subroutines.

   file is the name of the indirect-command file you want to run.

   This command runs the indirect-command file you specified, which in turn runs the FORTRAN test program. If more than one FORTRAN compiler is installed, the command file asks which one you want to use. You must specify the one with its OTS in SYSLIB. If you are using FORTRAN IV, the command file asks if your system has a floating point unit. If you are using FORTRAN 77, the question is not asked since you always have a floating-point unit.

   If your software is sound and you have performed all the above instructions correctly, the test program runs to completion and types some data on your terminal followed by the message

   TTn -- STOP file successful!

   where: file is the name of the indirect-command file you ran.

B.3.3 Error Conditions

If an indirect-command file finds an error, it stops running and types an error message on your terminal. However, this does not necessarily mean that your software is deficient or that it cannot be verified. An indirect-command file can stop running because some requirement has not been met even though your software is sound.

Some typical conditions that can cause error messages follow. If one of the indirect-command files will not run, and you get an error message, check to see that you have not neglected one of these conditions.

- A necessary system program such as the FORTRAN IV compiler, FOR.TSK, the FORTRAN 77, compiler, F77.TSK, or a utility program such as the RSX-11M task-builder, TKB.TSK, or PIP is not on your system device.
- You assigned the input device incorrectly. That is, the device you assigned to be the input device does not contain the volume where the Scientific Subroutines files that you will use as input reside.
- You assigned the output device incorrectly. That is, the device you assigned to be the output device does not have enough storage space.
- Your system device or your input or output device was write-protected when you tried to run an indirect-command file.

It is impossible to anticipate all conditions that can cause errors that are not the fault of the software. If you receive an error message, first check to see that all the requirements listed in Section B.1 are true for your system. Then, if you need to, see the appropriate RSX-11M/M-PLUS documentation to help you determine the cause of the error and correct it. You find the appropriate documentation by noting the name of the task which gave you the error message, for example, MCR, DSC, FLX, or FORTRAN. Then look in the RSX-11M/M-PLUS manual that contains information about that task to see what the error message means.

Finally, if one of the indirect-command files will not run, even though you have not made any of the errors described above or neglected any of the requirements listed in Section B.1, you may have received a defective copy of your Scientific Subroutines software. Contact DIGITAL for more information.

B.4 STORING THE SCIENTIFIC SUBROUTINES

After determining that the Scientific Subroutines that you want to use are sound, copy them to your system volume or to the development volume where you store your FORTRAN programs, or place them in a library (see Section B.6).

B.5 CREATING A PROGRAM THAT CALLS THE SCIENTIFIC SUBROUTINES

To create a FORTRAN program that calls the Scientific Subroutines, do the following:

1. Write and check your program.

2. Use one of the RSX-11M/M-PLUS editors, such as EDI, to enter your program into a source file. Type:

   >EDI prog.FTN (RET)

   where: prog is the name of your FORTRAN source program.

   For information about entering the text of your file, making changes, and displaying the file, see the RSX-11M/M-PLUS Utilities Manual and the Introduction to RSX-11M and RSX-11M-PLUS.

   NOTE

   Always specify a file extension when you use an editor. RSX-11M/M-PLUS editors do not use default file extensions. In this case, give your source file the extension .FTN since that is the default extension the FORTRAN IV or FORTRAN 77 compiler uses when it compiles your program.

3. Use the FORTRAN IV or FORTRAN 77 compiler to create object files of your program and of the Scientific Subroutines you wish to use.

   If you are using the FORTRAN IV compiler, type:

   >FOR prog=prog (RET)

   >FOR sub=sub (RET)

   If you are using the FORTRAN 77 compiler, type:

   >F77 prog=prog (RET)

   >F77 sub=sub (RET)

   where: prog is the name of your FORTRAN source program.

   sub is the name of the Scientific Subroutine you wish to compile.

   Since you can specify only one input file each time you use the FORTRAN IV or FORTRAN 77 compilers under RSX-11M/M-PLUS, you must compile the subroutines one at a time.

   Frequently used Scientific Subroutines can be compiled in advance. Then you only need to compile the main program.

4. Use the RSX-11M/M-PLUS task-builder to task-build the object file of your program with the object files of your Scientific Subroutines. The /FP switch tells the task-builder to use the floating point unit. Use the /FP switch if you use FORTRAN IV and your system has a floating point unit. Always use the /FP switch if you use FORTRAN 77 since your system always has a floating-point unit. Type:

   ```
   >TKB prog[/FP]=prog,subl,sub2,...subn(RET)
   ```

   where: prog is the name of your FORTRAN program.

   subl is the name of the first Scientific Subroutine you want to task-build.

   subn is the name of the last Scientific Subroutine you want to task-build.

   You can specify as many object files as you wish for input. However, if your list of input files will use more than one line on your terminal, you should see the RSX-11M/M-PLUS Task Builder Manual for information about how to input files using multiple lines.

   To avoid task-building individual subroutines to your program, you can place the subroutines in a library. See Section B.6

5. Run the task. Type:

   ```
   >RUN prog(RET)
   ```

   where: prog is the name of your executable program.

For more information about compiling, linking, and running FORTRAN programs, see the Introduction to RSX-11M and RSX-11M-PLUS, the IAS/RSX-11 FORTRAN IV User's Guide, and the PDP-11 FORTRAN 77 User's Guide.

B.6 USING LIBRARIES

Once you decide which of the Scientific Subroutines you will use most frequently, you can task-build them to your programs more easily by placing them in a library. When you place them in a library, you do not need to list each individual subroutine in the TKB command line. You only need to list the library name. When you place them in the system library, SYSLIB.OLB, you do not even need to list the library name in the TKB command line. The RSX-11M/M-PLUS task-builder automatically searches the system library for a subroutine or function needed by a program.

For example, suppose your compiled program, MYPROG.OBJ, calls the compiled subroutines ABSNT.OBJ, BOUND.OBJ, and GAUSS.OBJ. To task-build MYPROG, you have to type:

```
>TKB MYPROG[/FP]=MYPROG,ABSNT,BOUND,GAUSS(RET)
```

If you place the subroutines in a library in your UIC, called, for example, MYLIB.OLB, you task-build MYPROG by typing:

```
>TKB MYPROG[/FP]=MYPROG,MYLIB/LB(RET)
```

You have to use the /LB switch to tell RSX-11M/M-PLUS that MYLIB is a library file. However, if you add the subroutines to the system library, SYSLIB.OLB, you task-build MYPROG by typing:

```
>TKB MYPROG[/FP]=MYPROG(RET)
```

To create a library from the Scientific Subroutines you compile, use the RSX librarian program, LBR. You may also use LBR to add modules to or delete modules from a library at a later time. This includes the system library, SYSLIB.OLB. See the RSX-11M/M-PLUS Utilities Manual for information about LBR.

APPENDIX C

ALPHABETICAL INDEX OF SUBROUTINES

This appendix contains an alphabetical list of the Scientific Subroutines with the approximate number of words of memory required for each object file, and a list of any subroutines and functions it calls.

Subroutines preceded by an asterisk may produce a warning message when compiled with the FORTRAN IV compiler. These warning messages can be ignored.

The file extension .ext is .FOR for subroutines in the RT-11 operating system and .FTN for subroutines in the RSX-11M operating system.

Subroutine	Approximate Object File Length	Subroutine Called
ABSNT.ext	79 words	
ARRAY.ext	142 words	
AUTO.ext	145 words	
AVCAL.ext	228 words	
AVDAT.ext	248 words	
BESI.ext	385 words	
BESJ.ext	393 words	
BESK.ext	659 words	
BESY.ext	690 words	
BOUND.ext	215 words	
*CADD.ext	83 words	uses LOC
CANOR.ext	814 words	uses MINV,NROOT,EIGEN
*CCPY.ext	73 words	uses LOC
*CCUT.ext	135 words	uses LOC
CEL1.ext	122 words	
CEL2.ext	183 words	
CHISQ.ext	314 words	
CINT.ext	79 words	
CORRE.ext	944 words	uses DATA (user)
CROSS.ext	207 words	
CS.ext	319 words	
CSRT.ext	257 words	uses LOC,CCPY
*CSUM.ext	84 words	uses LOC
*CTAB.ext	168 words	uses LOC,CADD
*CTIE.ext	140 words	uses LOC
*DCLA.ext	49 words	uses LOC
*DCPY.ext	53 words	uses LOC
DISCR.ext	698 words	
DMATX.ext	318 words	
EIGEN.ext	983 words	
EXPI.ext	372 words	
EXSMO.ext	200 words	

ALPHABETICAL INDEX OF SUBROUTINES

Subroutine	Approximate Object File Length	Subroutine Called
FORIF.ext	282 words	uses external function
FORIT.ext	267 words	
GAMMA.ext	241 words	
GAUSS.ext	61 words	
GDATA.ext	511 words	
GMADD.ext	49 words	
GMPRD.ext	117 words	
GMSUB.ext	49 words	
GMTRA.ext	64 words	
GTPRD.ext	112 words	
KRANK.ext	465 words	uses RANK,TIE
LEP.ext	94 words	
LOAD.ext	77 words	
LOC.ext	110 words	
*MADD.ext	235 words	uses LOC
*MATA.ext	181 words	uses LOC
MCPY.ext	54 words	uses LOC
MEANQ.ext	481 words	
MFUN.ext	58 words	uses LOC, external function
MINV.ext	702 words	
MOMEN.ext	352 words	
*MPRD.ext	210 words	uses LOC
*MSTR.ext	113 words	uses LOC
*MSUB.ext	235 words	uses LOC
MTRA.ext	88 words	uses LOC,MCPY
MULTR.ext	458 words	
NROOT.ext	544 words	uses EIGEN
ORDER.ext	171 words	
PADD.ext	120 words	
PADDM.ext	125 words	
PCLA.ext	47 words	
PCLD.ext	70 words	
PDER.ext	69 words	
PDIV.ext	171 words	uses PNORM
PGCD.ext	107 words	uses PDIV,PNORM
PILD.ext	57 words	uses PQSD
PINT.ext	71 words	
PMPY.ext	109 words	
PNORM.ext	41 words	
POLRT.ext	850 words	
PQSD.ext	69 words	
PSUB.ext	121 words	
PVAL.ext	49 words	
PVSUB.ext	138 words	uses PMPY,PADDM,PCLA
QATR.ext	341 words	uses external function
QSF.ext	624 words	
QTEST.ext	168 words	
*RADD.ext	83 words	uses LOC
RANK.ext	179 words	
*RCPY.ext	73 words	uses LOC
RCUT.ext	135 words	uses LOC
RECP.ext	49 words	
RINT.ext	80 words	
RKGS.ext	1040 words	uses 2 subroutines
*RK1.ext	465 words	uses external function
RK2.ext	196 words	uses external function
*RSRT.ext	278 words	uses LOC
*RSUM.ext	84 words	uses LOC
*RTAB.ext	168 words	uses LOC,RADD
*RTIE.ext	155 words	uses LOC

ALPHABETICAL INDEX OF SUBROUTINES

Subroutine	Approximate Object File Length	Subroutine Called
RTMI.ext	545 words	uses external function
RTNI.ext	163 words	uses subroutine
RTWI.ext	196 words	uses external function
SADD.ext	57 words	uses LOC
SCLA.ext	49 words	uses LOC
SCMA.ext	96 words	
SDIV.ext	67 words	uses LOC
SICI.ext	375 words	
SIMQ.ext	469 words	
SMO.ext	120 words	
SMPY.ext	37 words	uses LOC
SRANK.ext	324 words	uses RANK,TIE
SRMA.ext	97 words	
SSUB.ext	57 words	uses LOC
SUBMX.ext	100 words	
SUBST.ext	187 words	uses subroutine
TAB1.ext	526 words	
TAB2.ext	976 words	
TALLY.ext	326 words	
TIE.ext	164 words	
*TPRD.ext	207 words	uses LOC
TRACE.ext	124 words	
TTSTT.ext	477 words	
TWOAV.ext	246 words	uses RANK
UTEST.ext	260 words	uses RANK,TIE
VARMX.ext	1089 words	
WTEST.ext	370 words	uses RANK,TIE
*XCPY.ext	108 words	uses LOC

INDEX

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____ Telephone _____

Street _____

City _____ State _____ Zip Code _____
or Country

----- Do Not Tear – Fold Here and Tape -----

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MR1–2/L12
MARLBOROUGH, MASSACHUSETTS 01752

----- Do Not Tear – Fold Here and Tape -----

UPDATE NOTICE

Scientific Subroutines Programmer's Reference Manual

AD–1101D–T1

January 1982

Insert this Update Notice in the *Scientific Subroutines Programmer's Reference Manual* to maintain an up-to-date record of changes to the manual.

Changed Information

The changed pages contained in this update package reflect the change from Version 3.2 to Version 4.0 of the RSX–11M operating system.

The instructions for inserting this update start on the next page.

INSTRUCTIONS
AD–1101D–T1

The following list of page numbers specifies which pages are to be placed in the *Scientific Subroutines Programmer's Reference Manual* as replacements for, or additions to, current pages.

Title page Copyright page	B–3 B–4	B–13 B–14
vii viii	B–5 B–6	B–15 B–16
ix blank	B–7 B–8	B–17 B–18
B–1 B–2	B–9 B–10	Index–1 Index–2

KEEP THIS UPDATE NOTICE IN YOUR MANUAL TO MAINTAIN AN UP-TO-DATE RECORD OF CHANGES.

TYPE AND IDENTIFICATION OF DOCUMENTATION CHANGES.
Five types of changes are used to update documents contained in the RSX software manuals. Change symbols and notations are used to specify where, when, and why alterations were made to each update page. The five types of update changes and the manner in which each is identified are described in the following table.

The Following Symbols and/or Notations	Identify the Following Types of Update Changes
1. Change bar in outside margin; version number and change date printed at bottom of page.	1. Changes were required by a new version of the software being described.
2. Change bar in outside margin; change date printed at bottom of page.	2. Changes were required to either clarify or correct the existing material.
3. Change date printed at bottom of page.	3. Changes were made for editorial purposes but use of the software is not affected.
4. Bullet (•) in outside margin; version number and change date printed at bottom of page.	4. Data was deleted to comply with a new version of the software being described.
5. Bullet (•) in outside margin; change date printed at bottom of page.	5. Data was deleted to either clarify or correct the existing material.