IBM

Application Program

H20-0252-3

1130 Scientific Subroutine Package

(1130-CM-02X)

Programmer's Manual

The Scientific Subroutine Package (SSP) is a collection of 121 FORTRAN subroutines divided, for the sake of presentation, into three groups: statistics, matrix manipulation, and other mathematics. It is a collection of input/output-free computational building blocks that can be combined with a user's input, output, or computational routines to meet his needs. The package can be applied to the solution of many problems in industry, science, and engineering.

Kristofer Sweger

Fourth Edition (September 1968)

This is a major revision obsoleting H20-0252-2. The storage requirements in Appendix A have been changed. Other changes are indicated by a vertical line to the left of the text.

This edition applies to Version 1, Modification 2 of the 1130 Scientific Subroutine Package (1130-CM-02X) and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters.

Changes are continually made to the specifications herein. Before using this publication in connection with the operation of IBM systems, consult the latest 1130 SRL Newsletter, N20-1130, for the editions that are applicable and current.

Copies of this and other IBM publications can be obtained through IBM branch offices. Address comments concerning the contents of this publication to IBM, Technical Publications Department, 112 East Post Road, White Plains, N.Y. 10601

CONTENTS

INTRODUCTION

The IBM 1130 Scientific Subroutine Package makes available a mathematical and statistical subroutine library. The user may supplement or modify the collection to meet his needs. This library includes a wide variety of subroutines to perform the functions listed below, but is not intended to be exhaustive in terms of either functions performed or methods used.

AREAS OF APPLICATION

Individual subroutines, or a combination of them, can be used to carry out the listed functions in the following areas:

Statistics

- Analysis of variance (factorial design)
- Correlation analysis
- Multiple linear regression
- Polynomial regression
- Canonical correlation
- Factor analysis (principal components, varimax)
- Discriminant analysis (many groups)
- Time series analysis
- Data screening and analysis
- Nonparametric tests
- Random number generation (uniform, normal)

Matrix Manipulation

- Inversion
- Eigenvalues and eigenvectors (real symmetric case)
- Simultaneous linear algebraic equations
- Transpositions
- Matrix arithmetic (addition, product, etc.)
- Partitioning
- Tabulation and sorting of rows or columns
- Elementary operations on rows or columns

Other Mathematical Areas

- Integration of given or tabulated functions
- Integration of first-order differential equations
- Fourier analysis of given or tabulated functions
- Bessel and modified Bessel function evaluation
- Gamma function evaluation
- Legendre function evaluation
- Elliptic, exponential, sine, cosine, Fresnel integrals
- Finding real roots of a given function
- Finding real and complex roots of a real polynomial
- Polynomial arithmetic (addition, division, etc.)
- Polynomial evaluation, integration, differentiation

CHARACTERISTICS

Some of the characteristics of the Scientific Subroutine Package are:

- All subroutines are free of input/output statements.
- Subroutines do not contain fixed maximum dimensions for the data arrays named in their calling sequences.
- All subroutines are written in 1130 FORTRAN.
- Many matrix manipulation subroutines handle symmetric and diagonal matrices (stored in economical, compressed formats) as well as general matrices. This can result in considerable saving in data storage for large arrays.
- The use of the more complex subroutines (or groups of them) is illustrated in the program documentation by sample main programs with input/output.
- All subroutines are documented uniformly.

DESIGN PHILOSOPHY

CHOICE OF ALGORITHMS

The algorithms in SSP have been chosen after considering questions of storage, accuracy, and past experience with the algorithm. Conservation of storage has been the primary criterion except in those situations where other considerations outweighed that of storage. As a result, many compromises have been made both with respect to level of sophistication and execution time. One such compromise is the use of the Runge-Kutta integration technique rather than predictor-corrector methods. A departure from the primary criterion of storage is illustrated by the algorithm for matrix inversion. If only row pivoting had been used, the subroutine would not have required working storage and would have needed fewer FORTRAN statements for implementation. However, pivoting on both rows and columns was chosen because of the accuracy requirement for matrix inversion in statistical operations.

PROGRAMMING

The subroutines in SSP have been programmed in 1130 FORTRAN. Many of the larger functions such as those in statistics have been programmed as a series or sequence of subroutines.

An example of the use of sequences of subroutines is the statistical function called factor analysis. Factor analysis is a method of analyzing the intercorrelations within a set of variables. It determines whether the variance in the original set of variables can be accounted for adequately by a smaller number of basic categories; namely, factors. In the Scientific Subroutine Package, factor analysis is normally performed by calling the following five subroutines in sequence:

1. CORRE - to find means, standard deviations, and correlation matrix
2. EIGEN - to compute eigenvalues and associated eigenvectors of the correlation matrix
3. TRACE - to select the eigenvalues that are greater than or equal to the control value specified by the user
4. LOAD - to compute a factor matrix
5. VARMX - to perform varimax rotation of the factor matrix

The multiple use of subroutines is illustrated by the fact that subroutine CORRE is also utilized in the multiple linear regression and canonical correlation. Subroutine EIGEN is used in canonical correlation as a third level subroutine.

OVERALL RULES OF USAGE

GENERAL RULES

All subroutines in the Scientific Subroutine Package (SSP) are entered by means of the standard FORTRAN CALL statement. These subroutines are purely computational in nature and do not contain any references to input/output devices. The user must therefore furnish, as part of his program, whatever input/output and other operations are necessary for the total solution of his problem. In addition, the user must define by DIMENSION statements all matrices to be operated on by SSP subroutines as well as those matrices utilized in his program. The subroutines contained in SSP are no different from any user-supplied subroutine. All of the normal rules of FORTRAN concerning subroutines must, therefore, be adhered to with the exception that the dimensioned areas in the SSP subroutine are not required to be the same as those in the calling program.

The CALL statement transfers control to the subroutine and replaces the dummy variables in that subroutine with the value of the actual arguments that appear in the CALL statement if the argument is a constant or a variable. When the argument is an array or function subprogram name, the address of the array or subprogram is transmitted to the called subroutine.

The arguments in a CALL statement must agree in order, number, and type with the corresponding arguments in the subroutine. A number may be passed to a subroutine either as a variable name in the argument list or as a constant in the argument list. For example, if the programmer wishes to add matrix AR1 to matrix AR2 in order to form matrix AR3 using the SSP subroutine GMADD and if AR1 and AR2 are both matrices with ten rows and twenty columns, either of the two following methods could be used:

Method 1

```
      .
CALL GMADD(AR1,AR2,AR3,10,20)
      .
```

Method 2

```
      .
N = 10
M = 20
CALL GMADD(AR1,AR2,AR3,N,M)
      .
```

Many of the subroutines in SSP require the name of a user function subprogram or a FORTRAN-supplied function name as part of the argument list in the CALL statement. If the user's program contains such a CALL, the function name appearing in the argument list must also appear in an EXTERNAL statement at the beginning of that program.

For example, the SSP subroutine RK2 integrates a function furnished by the user. It is therefore necessary for the user to program the function and give the name of the function to RK2 as a parameter in the CALL statement. If the user wished to integrate the function $\frac{dy}{dx} = 3.0x + 2.0Y$, his main program might look like:

```
      EXTERNAL DERY
            .
            .
      CALL RK2(DERY, .....)
            .
            .
      RETURN
      END
```

His function subprogram could be:

```
      FUNCTION DERY (X,Y)
      DERY=3.0*X+2.0*Y
      RETURN
      END
```

The user's main program gives the name of the programmed function to RK2 by including that name in the CALL statement and in an EXTERNAL statement. RK2, in turn, goes to the function DERY each time it requires a value for the derivative. The subroutine RK2 is not modified by the programmer. The dummy function name FUN in subroutine RK2 is, in effect, replaced by the name appearing in the user's CALL statement during execution of the subroutine.

MATRIX OPERATIONS

Special consideration must be given to the subroutines that perform matrix operations. These subroutines have two characteristics that affect the format of the data in storage--variable dimensioning and data storage compression.

Variable Dimensioning

Those subroutines that deal with matrices can operate on any size array limited, in most cases, only by the available core storage and numerical analysis considerations. The subroutines do not contain fixed maximum dimensions for data arrays named in their calling sequence. The variable dimension capability

has been implemented in SSP by using a vector storage approach. Under this approach, each column of a matrix is immediately followed in storage by the next column. Vector storage and two-dimensional storage result in the same layout of data in core, so long as the number of rows and columns in the matrix are the same as those in the user's dimension statement. If, however, the matrix is smaller than the dimensioned area, the two forms of storage are not compatible.

Consider the layout of data storage when operating on a 5 by 5 array of numbers in an area dimensioned as 10 by 10. If the programmer has been using double subscripted variables in the normal FORTRAN sense, the 25 elements of data will appear as shown in Figure 1. FORTRAN stores double subscripted data by column based on the column length specified in the DIMENSION statement. Thus, in the example, sequential core locations would contain data elements 1 to 5, five blank locations, data elements 6 to 10, five blank locations, etc. The matrix subroutines take a vector approach in storing arrays by column, which means that they assume the data is stored as shown in Figure 2.

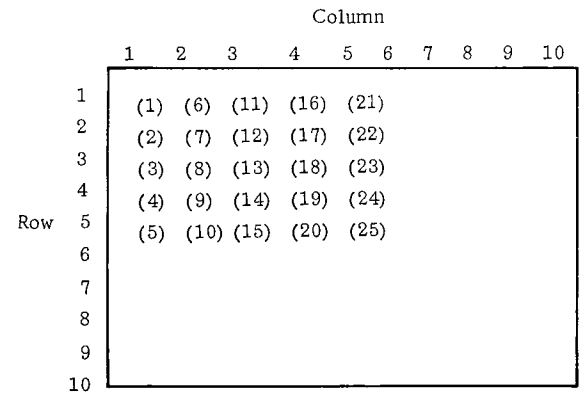


Figure 1. Double subscripted data storage

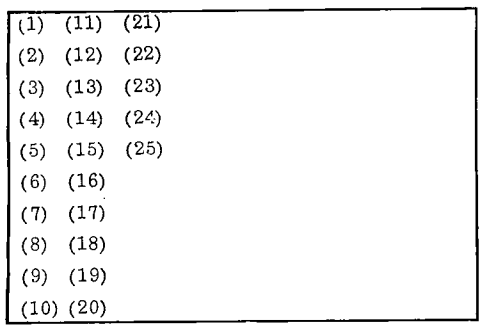


Figure 2. Vector storage

As has been stated previously, for the case where the dimensioned area is the same as the matrix size, the two approaches will have the same data storage layout and the user can proceed in a regular double subscripted fashion. If, however, he is operating in a mode where the dimensioned area is larger than the arrays and if he wishes to use the SSP subroutines, he must be certain that his data is stored in the vector fashion illustrated by Figure 2. A subroutine called ARRAY is available in SSP to change from one form of storage to the other. In addition, a subroutine called LOC is available to assist in referencing elements in an array stored in the vector fashion.

Storage Compression

Many subroutines in SSP can operate on compressed forms of matrices, as well as the normal form. Using this capability, which is called "storage mode", considerable savings in data storage can be obtained for special forms of large arrays. The three modes of storage are termed general, symmetric, and diagonal. In this context, general mode is one in which all elements of the matrix are in storage. Symmetric mode is one in which only the upper triangular portion of the matrix is retained columnwise in sequential locations in storage. (The assumption is made that the corresponding elements in the lower triangle have the same value.) Diagonal mode is one in which only the diagonal elements of the matrix are retained in sequential locations in storage. (The off-diagonal elements are assumed to be zero.) This capability has been implemented using the vector storage approach. To illustrate the effect of the storage mode capability, refer to Figure 3. A symmetric matrix is shown in Figure 3A. If this array is to be manipulated using the SSP matrix subroutines with storage mode capability, then the array may be stored as shown in Figure 3B. This is the upper triangular portion of the array and corresponds to a storage mode code of 1. Symmetric matrices of order N may be stored in a vector only N*(N+1)/2 locations rather than N*N locations. For larger matrices, this will be a saving of almost one half.

The effect of storage mode when dealing with diagonal matrices is even more pronounced. Diagonal matrices of order N may be stored in a vector only N locations long. Figure 3C shows a 3 by 3 diagonal matrix. If this array is to be manipulated using the SSP matrix subroutines with storage mode capability, then only the diagonal elements of the array need be stored. This is shown in Figure 3D and corresponds to a storage mode code of 2.

General matrices of order N by M require a vector N*M long and use a storage mode code of 0.

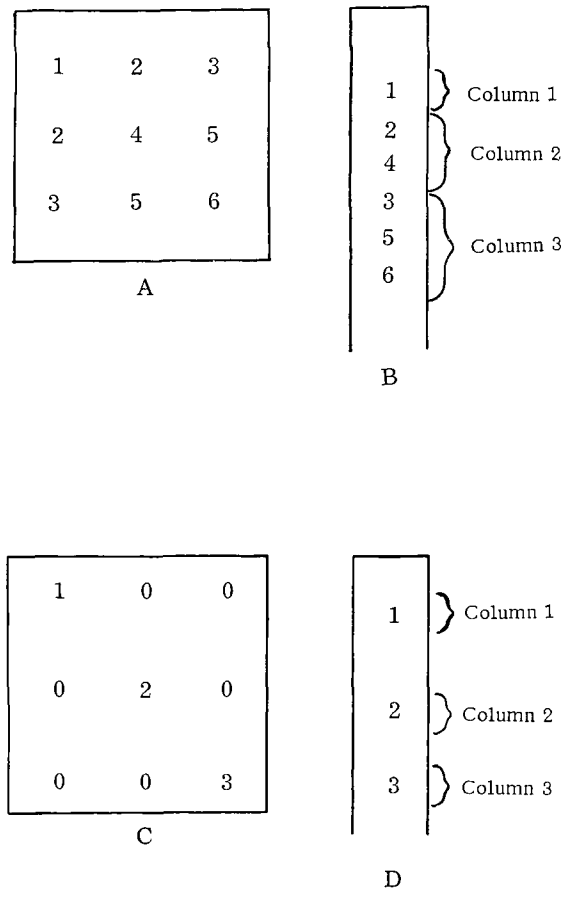


Figure 3. Storage mode

Thus, if the programmer wishes to use SSP subroutines on matrix A, which is general, matrix B, which is symmetric, and matrix C, which is diagonal, and all matrices are 10 by 10 or smaller, the dimension statement in his program could be:

```
DIMENSION A(100), B(55), C(10)
```

Matrix Element References

Subroutine LOC in the Scientific Subroutine Package may be used to reference elements within a matrix that is stored in a vector fashion and may involve storage mode compression. The calling sequence for LOC is:

```
CALL LOC (I, J, IJ, N, M, MS)
```

The capabilities of subroutine LOC are as follows: If reference is required to the element at row I and column J of matrix A whose dimensions are N by M and if the storage mode code is MS, then a CALL to the LOC subroutine as shown above will result in the computation of the subscript IJ such that A(IJ) is the desired element. The parameters represented by I, J, N, M, MS can either be integer variables or integer constants. The parameter represented by IJ is an integer variable. Note that the user must dimension the array A as a single subscripted variable to meet the restrictions of some FORTRAN systems. To illustrate the use of LOC: If reference is required to the element at row 2, column 2 of the 3 by 3 symmetric matrix illustrated in Figure 3A and stored as shown in Figure 3B (storage mode code 1), the sequence might be:

```
CALL LOC (2, 2, IJ, 3, 3, 1)
```

The value of IJ computed by LOC would be 3; meaning that the proper element is the third element in the specially stored symmetric matrix (Figure 3B). If the storage mode code is for a symmetric matrix where only the upper triangular portion is retained in storage and if I and J refer to an element in the lower triangular portion, IJ will contain the subscript for the corresponding element in the retained upper triangle. Thus if the user wanted the element in row 3, column 1 of the matrix shown in Figure 3A and the array was stored as in Figure 3B, the statement:

```
CALL LOC (3, 1, IJ, 3, 3, 1)
```

would result in IJ having the value of 4; that is, the fourth element in Figure 3B. If a matrix is stored as shown in Figure 3D (storage mode 2) and LOC is used to compute the subscript for an off-diagonal element (I not equal to J), the result in IJ will be zero. This is due to the fact that the element does not exist in storage. In this situation, the user must not utilize IJ as a subscript. Following is an illustration of how to take care of this condition and also handle the case where the current storage mode is unknown.

If the user wishes to set a variable X equal to the element in the third row and fourth column of a 10 by 10 array named A for either a symmetric, diagonal, or general matrix, the required program can be implemented for any storage mode MS as follows:

```
      CALL LOC (3, 4, IJ, 10, 10, MS)
      X = 0.0
      IF(IJ)20, 30, 20
   20 X = A(IJ)
   30 --------
```

MS is assumed to have been set at 0, 1, or 2 at some earlier point in the program. This sequence would then set the proper value for X given any storage mode that might be encountered. The second and third statements take care of the off-diagonal condition for a matrix with a storage mode of 2.

As a special case, LOC can be used to compute the total length of an array in storage with a statement such as:

CALL LOC (N, M, IJ, N, M, MS)

For example, if the user has a 3 by 3 matrix whose storage mode is 1 (Figure 3B), the statement:

CALL LOC (3, 3, IJ, 3, 3, 1)

will result in IJ being set to 6. This is not only the proper subscript to reference element 3, 3 but is also the actual length of the vector in storage.

The information contained in the fifth parameter (number of columns) in the calling sequence for LOC is not actually used in the calculations performed by LOC. It has been included in the calling sequence in case the user wishes to expand LOC to cover other forms of data storage.

OPTIMIZATION OF TIME

The subroutines in SSP are designed to conserve storage. If the user wishes to exchange space for time, there are several ways in which SSP may be modified to effect this end. For example, many of the subroutines in SSP make use of LOC subroutine to handle vector storage and storage mode referencing. The execution time of these subroutines can be substantially reduced by implementing LOC in Assembler Language. (The distributed version of LOC is implemented in FORTRAN.) Another approach is to incorporate the function of LOC within each subroutine and thus avoid the "setup" costs of repeated calls to LOC. This has the effect of reducing execution time but at some cost in subroutine storage and in the ease with which other modes of storage such as triangular matrix storage or storage by row rather than by column can be implemented. Figure 4 shows how matrix addition and the LOC capabilities can be implemented within the same subroutine.

In the mathematical area, the user may find it desirable to implement entirely different algorithms for integration. The use of techniques that automatically adjust the integration interval depending on the rate of change of the function will often have the effect of reducing total execution time.

EXTENDED PRECISION

The accuracy of the computations in many of the SSP subroutines is highly dependent upon the number of significant digits available for arithmetic operations. Matrix inversion, integration, and many of the statistical subroutines fall into this category. All of the subroutines will compile correctly for extended precision by placing the *EXTENDED PRECISION control card at the appropriate place in the deck. Note that 1130 FORTRAN does not allow the intermixing of regular and extended precision in the same program.

```
      SUBROUTINE MADX(A,B,R,N,M,MSA,MSB)
      DIMENSION A(1),B(1),R(1)
C
C        TEST FOR SAME STORAGE MODE
C
      IF(MSA-MSB) 30,10,30
C
C        COMPUTE VECTOR LENGTH
C
   10 ND=N*M
      IF(MSA-1) 24,22,23
   22 ND=(ND+N)/2
      GO TO 24
   23 ND=N
C
C        ADD MATRICES OF SAME STORAGE MODE
C
   24 DO 25 I=1,ND
   25 R(I)=A(I)+B(I)
      RETURN
C
C        GET STORAGE MODE OF OUTPUT MATRIX
C
   30 MTEST=MSA*MSB
      MSR=0
      IF(MTEST) 35,35,32
   32 MSR=1
   35 DO 60 J=1,M
      DO 60 I=1,N
C
C        LOCATE ELEMENT IN OUTPUT MATRIX
C
      KX=-1
      MS=MSR
      GO TO 65
   40 IJR=IR
C
C        LOCATE ELEMENT IN MATRIX A
C
      KX=0
      MS=MSA
      GO TO 65
   45 IJA=IR
      AEL=0.0
      IF(IJA) 46,48,46
   46 AEL=A(IJA)
C
C        LOCATE ELEMENT IN MATRIX B
C
   48 KX=1
      MS=MSB
      GO TO 65
   50 IJB=IR
      BEL=0.0
      IF(IJB) 55,60,55
   55 BEL=B(IJB)
C
C        ADD MATRICES OF DIFFERENT STORAGE MODES
C
   60 R(IJR)=AEL+BEL
      RETURN
C
C        IN LINE LOC
C
   65 IF(MS-1) 70,75,90
   70 IR=N*(J-1)+I
      GO TO 95
   75 IF(I-J) 80,85,85
   80 IR=I+(J*J-J)/2
      GO TO 95
   85 IR=J+(I*I-I)/2
      GO TO 95
   90 IR=0
      IF(I-J) 95,92,95
   92 IR=I
   95 IF(KX) 40,45,50
      END
```

Figure 4. Inline LOC

FORMAT OF THE DOCUMENTATION

The major portion of this manual consists of the documentation for the individual subroutines and the sample programs.

SUBROUTINE DESCRIPTIONS

A guide to the subroutines, designed to aid in locating any particular subroutine, is given in the pages that follow. Each of the subroutine descriptions contains a program listing and, in some cases, a mathematical description. If there are restrictions on the ranges of values that the parameters may take, these are included under the remarks section of each subroutine description. References to books and periodicals will be found under the method section of the description. The mathematical description pages do not, in most cases, indicate the derivation of the mathematics. They are intended to indicate what mathematical operations are actually being performed in the subroutines. Some of the major statistical functions are performed by a sequence of SSP subroutines. An abstract describing this sequence will be found just before the description of the first subroutine that is specific to this function.

SAMPLE PROGRAM DESCRIPTIONS

The sample program listings are given in Appendix D. They are immediately preceded by a guide to aid in locating the sample program calling a particular SSP subroutine or (where applicable) typical user-written subroutine. Each sample program consists of a detailed description including information on the problem, the program, input, output, program modification, operating instructions, error messages, and machine listings of the programs, input data and output results. Timings for these programs is given in Appendix C. The sample programs have been chosen to (1) illustrate a sequence of SSP subroutines, (2) illustrate the use of a complex subroutine, or (3) show the way in which one member of a large set of related subroutines might be used.

As part of the development of the sample programs, some special sample subroutines have been implemented that may prove useful to the programmer. These include:

HIST - Print a histogram of frequencies

MATIN - Read an input matrix into storage in vector form for use by SSP matrix subroutines

PLOT - Plot several variables versus a base variable

MXOUT - Print a matrix stored in the SSP vector format

Listings of the above subroutines are included in the sample program documentation in this manual.

The sample programs all require 8K words of core for execution and several of them require (in addition) the overlay capabilities of the Disk Monitor.

MACHINE CONFIGURATION

The machine configuration necessary to run SSP/1130 is dependent upon the use that is to be made of the package. All of the subroutines are I/O free, compile to less than 1500 words of core, and are, therefore, configuration independent. However, many of the routines are intended to be used in conjunction with other subroutines or to solve problems using large arrays of data. For this reason, many of the subroutines are not useful with less than 8K words of core.

The following items should be taken into consideration when deciding upon the applicability of this package to a particular machine configuration:

1. The size of problem which may be executed on a given 1130 depends upon the number of subroutines used, the size of the compiled subroutines, the size of the compiled main program, the size of the control program, and the data storage requirements.
2. SSP/1130 programs will be distributed in card form only.
3. Several of the sample problems require 8K words of core and the use of the Disk Monitor, and the remaining sample problems require 8K words of core.

It is possible to estimate program sizes by using the manual Core Requirements for 1130 FORTRAN (C20-1641) in conjunction with the core size listing found in Appendix A.

GUIDE TO SUBROUTINES

The following pages give the subroutine listings. Wherever necessary, additional explanatory matter on the routine, or a discussion of the underlying mathematics has been included.

Statistics - Data Screening

TALLY

Purpose:

Calculate total, mean, standard deviation, minimum, maximum for each variable in a set (or a subset) of observations.

Usage:

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

Description of parameters:

- A - Observation matrix, NO by NV
- S - Input vector indicating subset of A. Only those observations with a non-zero S(J) are considered. Vector length is NO.
- TOTAL - Output vector of totals of each variable. Vector length is NV.
- AVER - Output vector of averages 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.

Remarks:

None.

Subroutines and function subprograms required:

None.

Method:

All observations corresponding to a non-zero element in S vector are analyzed for each variable in matrix A. Totals are accumulated and minimum and maximum values are found. Following this, means and standard deviations are calculated. The divisor for standard deviation is one less than the number of observations used.

```
      SUBROUTINE TALLY(A,S,TOTAL,AVER,SD,VMIN,VMAX,NO,NV)                TALLY  1
      DIMENSION A(1),S(1),TOTAL(1),AVER(1),SD(1),VMIN(1),VMAX(1)         TALLY  2
C        CLEAR OUTPUT VECTORS AND INITIALIZE VMIN,VMAX                   TALLY  3
      DO 1 K=1,NV                                                        TALLY  4
      TOTAL(K)=0.0                                                       TALLY  5
      AVER(K)=0.0                                                        TALLY  6
      SD(K)=0.0                                                          TALLY  7
      VMIN(K)=1.0E38                                                     TALLY  8
    1 VMAX(K)=-1.0E38                                                    TALLY  9
C        TEST SUBSET VECTOR                                              TALLY 10
      SCNT=0.0                                                           TALLY 11
      DO 7 J=1,NO                                                        TALLY 12
      IJ=J-NO                                                            TALLY 13
      IF(S(J)) 2,7,2                                                     TALLY 14
    2 SCNT=SCNT+1.0                                                      TALLY 15
C        CALCULATE TOTAL, MINIMA, MAXIMA                                 TALLY 16
      DO 6 I=1,NV                                                        TALLY 17
      IJ=IJ+NO                                                           TALLY 18
      TOTAL(I)=TOTAL(I)+A(IJ)                                            TALLY 19
      IF(A(IJ)-VMIN(I)) 3,4,4                                            TALLY 20
    3 VMIN(I)=A(IJ)                                                      TALLY 21
    4 IF(A(IJ)-VMAX(I)) 6,6,5                                            TALLY 22
    5 VMAX(I)=A(IJ)                                                      TALLY 23
    6 SD(I)=SD(I)+A(IJ)*A(IJ)                                            TALLY 24
    7 CONTINUE                                                           TALLY 25
C        CALCULATE MEANS AND STANDARD DEVIATIONS                         TALLY 26
      DO 8 I=1,NV                                                        TALLY 27
      AVER(I)=TOTAL(I)/SCNT                                              TALLY 28
    8 SD(I)=SQRT(ABS((SD(I)-TOTAL(I)*TOTAL(I)/SCNT)/(SCNT-1.0)))         TALLY 29
      RETURN                                                             TALLY 30
      END                                                                TALLY 31
```

BOUND

Purpose:

Select from a set (or a subset) of observations the number of observations under, between and over two given bounds for each variable.

Usage:

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

Description of parameters:

- A - Observation matrix, NO by NV
- S - Vector indicating subset of A. Only those observations with a non-zero S(J) are considered. Vector length is NO.
- BLO - Input vector of lower bounds on all variables. Vector length is NV.
- BHI - Input vector of upper bounds on all variables. Vector length is NV.
- UNDER - Output vector indicating, for each variable, number of observations under lower bounds. Vector length is NV.
- BETW - Output vector indicating, for each variable, number of observations equal to or between lower and upper bounds. Vector length is NV.
- OVER - Output vector indicating, for each variable, number of observations over upper bounds. Vector length is NV.
- NO - Number of observations
- NV - Number of variables for each observation

Remarks:

None.

Subroutines and function subprograms required:

None.

Method:

Each row (observation) of matrix A with corresponding non-zero element in S vector is tested. Observations are compared with specified lower and upper variable bounds and a count is kept in vectors under, between, and over.

```
      SUBROUTINE BOUND(A,S,BLO,BHI,UNDER,BETW,OVER,NO,NV)                BOUND  1
      DIMENSION A(1),S(1),BLO(1),BHI(1),UNDER(1),BETW(1),OVER(1)         BOUND  2
C        CLEAR OUTPUT VECTORS.                                           BOUND  3
      DO 1 K=1,NV                                                        BOUND  4
      UNDER(K)=0.0                                                       BOUND  5
      BETW(K)=0.0                                                        BOUND  6
    1 OVER(K)=0.0                                                        BOUND  7
C        TEST SUBSET VECTOR                                              BOUND  8
      DO 8 J=1,NO                                                        BOUND  9
      IJ=J-NO                                                            BOUND 10
      IF(S(J)) 2,8,2                                                     BOUND 11
C        COMPARE OBSERVATIONS WITH BOUNDS                                BOUND 12
    2 DO 7 I=1,NV                                                        BOUND 13
      IJ=IJ+NO                                                           BOUND 14
      IF(A(IJ)-BLO(I)) 5,3,3                                             BOUND 15
    3 IF(A(IJ)-BHI(I)) 4,4,6                                             BOUND 16
C        COUNT                                                           BOUND 17
    4 BETW(I)=BETW(I)+1.0                                                BOUND 18
      GO TO 7                                                            BOUND 19
    5 UNDER(I)=UNDER(I)+1.0                                              BOUND 20
      GO TO 7                                                            BOUND 21
    6 OVER(I)=OVER(I)+1.0                                                BOUND 22
    7 CONTINUE                                                           BOUND 23
    8 CONTINUE                                                           BOUND 24
      RETURN                                                             BOUND 25
      END                                                                BOUND 26
```

SUBST

Purpose:

Derive a subset vector indicating which observations in a set have satisfied certain conditions on the variables.

Usage:

CALL SUBST (A, C, R, B, S, NO, NV, NC)
Parameter B must be defined by an external statement in the calling program.

Description of parameters:

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 a relational code as follows:

1. for less than
2. for less than or equal to
3. for equal to
4. for not equal to
5. for greater than or equal to
6. for greater than

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

2.
5.
92.5

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

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 to be supplied by the user. It consists of a Boolean expression linking the intermediate values stored in vector R. The Boolean operators are '*' for 'and', '+' for 'or'. 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 non-zero. If it is not, S(I) is zero. Vector length is NO.

NO - Number of observations.

NV - Number of variables.

NC - Number of basic conditions to be satisfied.

Subroutines and function subprograms required:

B The name of actual subroutine supplied by the user may be different (e.g., BOOL), but subroutine SUBST always calls it as B. In order for subroutine SUBST to do this, the name of the user-supplied subroutine must be defined by an EXTERNAL statement in the calling program. The name must also be listed in the "CALL SUBST" statement. (See usage above.)

Method:

The following is done for each observation. Condition matrix is analyzed to determine which variables are to be examined. Intermediate vector R is formed. The Boolean expression (in subroutine B) is then evaluated to derive the element in subset vector S corresponding to the observation.

```
      SUBROUTINE SUBST(A,C,R,B,S,NO,NV,NC)                              SUBST  1
      DIMENSION A(1),C(1),R(1),S(1)                                     SUBST  2
      DO 9 I=1,NO                                                       SUBST  3
      IQ=I-NO                                                           SUBST  4
      K=-2                                                              SUBST  5
      DO 8 J=1,NC                                                       SUBST  6
C         CLEAR R VECTOR                                                SUBST  7
      R(J)=0.0                                                          SUBST  8
C         LOCATE ELEMENT IN OBSERVATION MATRIX AND RELATIONAL CODE      SUBST  9
      K=K+3                                                             SUBST 10
      IZ=C(K)                                                           SUBST 11
      IA=IQ+IZ*NO                                                       SUBST 12
      IGO=C(K+1)                                                        SUBST 13
C         FORM R VECTOR                                                 SUBST 14
      Q=A(IA)-C(K+2)                                                    SUBST 15
      GO TO(1,2,3,4,5,5),IGO                                            SUBST 16
    1 IF(Q) 7,8,8                                                       SUBST 17
    2 IF(Q) 7,7,8                                                       SUBST 18
    3 IF(Q) 8,7,8                                                       SUBST 19
    4 IF(Q) 7,8,7                                                       SUBST 20
    5 IF(Q) 8,7,7                                                       SUBST 21
    6 IF(Q) 8,8,7                                                       SUBST 22
    7 R(J)=1.0                                                          SUBST 23
    8 CONTINUE                                                          SUBST 24
C         CALCULATE S VECTOR                                            SUBST 25
    9 CALL B(R,S(I))                                                    SUBST 26
      RETURN                                                            SUBST 27
      END                                                               SUBST 28
```

ABSNT

Purpose:

Test missing or zero values for each observation in matrix A.

Usage:

CALL ABSNT (A, S, NO, NV)

Description of parameters:

- A - Observation matrix, NO by NV.
- S - Output vector of length NO indicating the following codes for each observation:
  - 1 There is not a missing or zero value.
  - 0 At least one value is missing or zero.
- NO - Number of observations.
- NV - Number of variables for each observation.

Remarks:

None.

Subroutines and function subprograms required:

None.

Method:

A test is made for each row (observation) of the matrix A. If there is not a missing or zero value, 1 is placed in S(J). If at least one value is missing or zero, 0 is placed in S(J).

```
      SUBROUTINE ABSNT(A,S,NO,NV)                                   ABSNT  1
      DIMENSION A(1),S(1)                                           ABSNT  2
      DO 20 J=1,NO                                                  ABSNT  3
      IJ=J-NO                                                       ABSNT  4
      S(J)=1.0                                                      ABSNT  5
      DO 10 I=1,NV                                                  ABSNT  6
      IJ=IJ+NO                                                      ABSNT  7
      IF(A(IJ)) 10,5,10                                             ABSNT  8
    5 S(J)=0                                                        ABSNT  9
      GO TO 20                                                      ABSNT 10
   10 CONTINUE                                                      ABSNT 11
   20 CONTINUE                                                      ABSNT 12
      RETURN                                                        ABSNT 13
      END                                                           ABSNT 14
```

TAB1

This subroutine tabulates for a selected variable in an observation matrix, the frequencies and percent frequencies over class intervals. Interval size is computed as follows:

$$k = \frac{UBO_3 - UBO_1}{UBO_2 - 2} \tag{1}$$

where UBO_1 = given lower bound

UBO_2 = given number of intervals

UBO_3 = given upper bound

If $UBO_1 = UBO_3$, the subroutine finds and uses the minimum and maximum values of the variable.

A table lookup is used to obtain the frequency of the i-th class interval for the variable, where $i = 1, 2, \ldots, UBO_2$. Then, each frequency is divided by the number of observations, n, to obtain the percent frequency:

$$P_i = \frac{100F_i}{n} \tag{2}$$

In addition, the following statistics are calculated for the variable:

$$\text{Total: } T = \sum_{i=1}^{n} X_{ij} \tag{3}$$

where j = selected variable

$$\text{Mean: } \bar{X} = \frac{T}{n} \tag{4}$$

Standard deviation:

$$s = \sqrt{\frac{\sum_{i=1}^{n} X_{ij}^2 - \left(\sum_{i=1}^{n} X_{ij}\right)^{2/n}}{n-1}} \tag{5}$$

Subroutine TAB1

Purpose:

Tabulate for one variable in an observation matrix (or a matrix subset), the frequency and percent frequency over given class intervals. In addition, calculate for the same variable the total, average, standard deviation, minimum, and maximum.

Usage:

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

Description of parameters:

- A - Observation matrix, NO by NV.
- S - Input vector giving subset of A. Only those observations with a corresponding non-zero 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, i.e., total, average, standard deviation, minimum and maximum. Vector length is 5.
- NO - Number of observations.
- NV - Number of variables for each observation.

Remarks:

None.

Subroutines and function subprograms required:

None.

Method:

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

```
      SUBROUTINE TAB1(A,S,NOVAR,UBO,FREQ,PCT,STATS,NO,NV)                TAB1   1
      DIMENSION A(1),S(1),UBO(3),FREQ(1),PCT(1),STATS(5)                 TAB1 M07
      DIMENSION WBO(3)                                                   TAB1 M01
      DO 5 I=1,3                                                         TAB1 M02
    5 WBO(I)=UBO(I)                                                      TAB1 M03
C        CALCULATE MIN AND MAX                                           TAB1   3
      VMIN=1.0E38                                                        TAB1   4
      VMAX=-1.0E38                                                       TAB1   5
      IJ=NO*(NOVAR-1)                                                    TAB1   6
      DO 30 J=1,NO                                                       TAB1   7
      IJ=IJ+1                                                            TAB1   8
      IF(S(J)) 10,30,10                                                  TAB1   9
   10 IF(A(IJ)-VMIN) 15,20,20                                            TAB1  10
   15 VMIN=A(IJ)                                                         TAB1  11
   20 IF(A(IJ)-VMAX) 30,30,25                                            TAB1  12
   25 VMAX=A(IJ)                                                         TAB1  13
   30 CONTINUE                                                           TAB1  14
      STATS(4)=VMIN                                                      TAB1  15
      STATS(5)=VMAX                                                      TAB1  16
C        DETERMINE LIMITS                                                TAB1  17
      IF(UBO(1)-UBO(3)) 40,35,40                                         TAB1  18
   35 UBO(1)=VMIN                                                        TAB1  19
      UBO(3)=VMAX                                                        TAB1  20
   40 INN=UBO(2)                                                         TAB1  21
C        CLEAR OUTPUT AREAS                                              TAB1  22
      DO 45 I=1,INN                                                      TAB1  23
      FREQ(I)=0.0                                                        TAB1  24
   45 PCT(I)=0.0                                                         TAB1  25
      DO 50 I=1,3                                                        TAB1  26
   50 STATS(I)=0.0                                                       TAB1  27
C        CALCULATE INTERVAL SIZE                                         TAB1  28
      SINT=ABS((UBO(3)-UBO(1))/(UBO(2)-2.0))                             TAB1  29
C        TEST SUBSET VECTOR                                              TAB1  30
      SCNT=0.0                                                           TAB1  31
      IJ=NO*(NOVAR-1)                                                    TAB1  32
      DO 75 J=1,NO                                                       TAB1  33
      IJ=IJ+1                                                            TAB1  34
      IF(S(J)) 55,75,55                                                  TAB1  35
   55 SCNT=SCNT+1.0                                                      TAB1  36
C        DEVELOP TOTAL AND FREQUENCIES                                   TAB1  37
      STATS(1)=STATS(1)+A(IJ)                                            TAB1  38
      STATS(3)=STATS(3)+A(IJ)*A(IJ)                                      TAB1  39
      TEMP=UBO(1)-SINT                                                   TAB1  40
      INTX=INN-1                                                         TAB1  41
      DO 60 I=1,INTX                                                     TAB1  42
      TEMP=TEMP+SINT                                                     TAB1  43
      IF(A(IJ)-TEMP) 70,60,60                                            TAB1  44
   60 CONTINUE                                                           TAB1  45
      IF(A(IJ)-TEMP) 75,65,65                                            TAB1  46
   65 FREQ(INN)=FREQ(INN)+1.0                                            TAB1  47
      GO TO 75                                                           TAB1  48
   70 FREQ(I)=FREQ(I)+1.0                                                TAB1  49
   75 CONTINUE                                                           TAB1  50
C        CALCULATE RELATIVE FREQUENCIES                                  TAB1  51
      DO 80 I=1,INN                                                      TAB1  52
   80 PCT(I)=FREQ(I)*100.0/SCNT                                          TAB1  53
C        CALCULATE MEAN AND STANDARD DEVIATION                           TAB1  54
      IF(SCNT-1.0) 85,85,90                                              TAB1  55
   85 STATS(2)=0.0                                                       TAB1  56
      STATS(3)=0.0                                                       TAB1  57
      GO TO 95                                                           TAB1  58
   90 STATS(2)=STATS(1)/SCNT                                             TAB1  59
      STATS(3)=SQRT(ABS((STATS(3)-STATS(1)*STATS(1)/SCNT)/(SCNT-1.0)))   TAB1  60
   95 DO 100 I=1,3                                                       TAB1 M04
  100 UBO(I)=WBO(I)                                                      TAB1 M05
      RETURN                                                             TAB1 M06
      END                                                                TAB1  62
```

TAB2

This subroutine performs a two-way classification of the frequency, percent frequency, and other statistics over given class intervals, for two selected variables in an observation matrix.

Interval size for each variable is computed as follows:

$$k_j = \frac{UBO_{3j} - UBO_{1j}}{UBO_{2j} - 2} \quad (1)$$

where UBO_{1j} = given lower bound

UBO_{2j} = given number of intervals

UBO_{3j} = given upper bound

$j = 1, 2$

If $UBO_{1i} = UBO_{3i}$, the subroutine finds and uses the minimum and maximum values of the j^{th} variable.

A frequency tabulation is then made for each pair of observations in a two-way table as shown in Figure 5.

Symbols $\geq$ and $<$ in Figure 5 indicate that a count is classified into a particular interval if the data point is greater than or equal to the lower limit of that interval but less than the upper limit of the same interval.

Then, each entry in the frequency matrix, F_{ij}, is divided by the number of observations, N, to obtain the percent frequency:

$$P_{ij} = \frac{100F_{ij}}{N} \quad (2)$$

where $i = 1, 2, \ldots, UBO_{21}$

$j = 1, 2, \ldots, UBO_{22}$

As data are classified into the frequency matrix, the following intermediate results are accumulated for each class interval of both variables:

1. Number of data points, n
2. Sum of data points, $\sum_{i=1}^{n} X_i$
3. Sum of data points squared, $\sum_{i=1}^{n} X_i^2$

From these, the following statistics are calculated for each class interval:

Mean: $$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n} \quad (3)$$

Standard deviation:

$$s = \sqrt{\frac{\sum_{i=1}^{n} X_i^2 - \left(\sum_{i=1}^{n} X_i\right)^2 / n}{n - 1}} \quad (4)$$

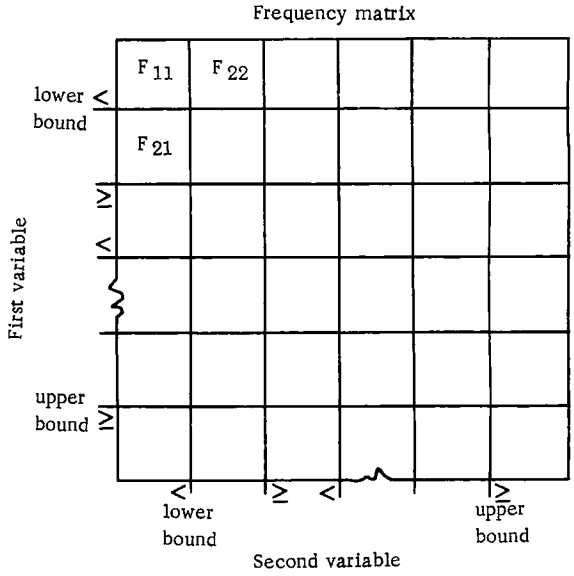


Figure 5. Frequency matrix

Subroutine TAB2

Purpose:

Perform a two-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.

Usage:

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

Description of parameters

- A - Observation matrix, NO by NV
- S - Input vector giving subset of A. Only those observations with a corresponding non-zero 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.

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 two-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.

NV - Number of variables for each observation.

Remarks:

None.

Subroutines and function subprograms required:

None.

Method:

Interval sizes for both variables are calculated from the given information or optionally from the minimum and maximum values. The frequency and percent frequency matrices are developed. Matrices STAT1 and STAT2 summarizing totals, means, and standard deviations are then calculated. The divisor for standard deviation is one less than the number of observations used in each class interval.

```
      SUBROUTINE TAB2(A,S,NOV,UBO,FREQ,PCT,STAT1,STAT2,NO,NV)          TAB2   1
      DIMENSION A(1),S(1),NOV(2),UBO(3,2),FREQ(1),PCT(1),STAT1(1),     TAB2   2
     1STAT2(2),SINT(2)                                                 TAB2   3
      DIMENSION WBO(3,2)                                               TAB2 M01
      DO 5 I=1,3                                                       TAB2 M02
      DO 5 J=1,2                                                       TAB2 M03
    5 WBO(I,J)=UBO(I,J)                                                TAB2 M04
C        DETERMINE LIMITS                                              TAB2   4
      DO 40 I=1,2                                                      TAB2 M05
      IF(UBO(1,I)-UBO(3,I))40,10,40                                    TAB2 M06
   10 VMIN=1.0E38                                                      TAB2 M07
      VMAX=-1.0E38                                                     TAB2   8
      IJ=NO*(NOV(I)-1)                                                 TAB2   9
      DO 35 J=1,NO                                                     TAB2  10
      IJ=IJ+1                                                          TAB2  11
      IF(S(J)) 15,35,15                                                TAB2  12
   15 IF(A(IJ)-VMIN) 20,25,25                                          TAB2  13
   20 VMIN=A(IJ)                                                       TAB2  14
   25 IF(A(IJ)-VMAX) 35,35,30                                          TAB2  15
   30 VMAX=A(IJ)                                                       TAB2  16
   35 CONTINUE                                                         TAB2  17
      UBO(1,I)=VMIN                                                    TAB2  18
      UBO(3,I)=VMAX                                                    TAB2 M08
   40 CONTINUE                                                         TAB2 M09
C        CALCULATE INTERVAL SIZE                                       TAB2  20
   45 DO 50 I=1,2                                                      TAB2  21
   50 SINT(I)=ABS((UBO(3,I)-UBO(1,I))/(UBO(2,I)-2.0))                  TAB2  22
C        CLEAR OUTPUT AREAS                                            TAB2  23
      INT1=UBO(2,1)                                                    TAB2  24
      INT2=UBO(2,2)                                                    TAB2  25
      INTT=INT1*INT2                                                   TAB2  26
      DO 55 I=1,INTT                                                   TAB2  27
      FREQ(I)=0.0                                                      TAB2  28
   55 PCT(I)=0.0                                                       TAB2  29
      INTY=3*INT1                                                      TAB2  30
      DO 60 I=1,INTY                                                   TAB2  31
   60 STAT1(I)=0.0                                                     TAB2  32
      INTZ=3*INT2                                                      TAB2  33
      DO 65 I=1,INTZ                                                   TAB2  34
   65 STAT2(I)=0.0                                                     TAB2  35
C        TEST SUBSET VECTOR                                            TAB2  36
      SCNT=0.0                                                         TAB2  37
      INTY=INT1-1                                                      TAB2  38
      INTX=INT2-1                                                      TAB2  39
      IJ=NO*(NOV(1)-1)                                                 TAB2  40
      IJX=NO*(NOV(2)-1)                                                TAB2  41
      DO 95 J=1,NO                                                     TAB2  42
      IJ=IJ+1                                                          TAB2  43
      IJX=IJX+1                                                        TAB2  44
      IF(S(J)) 70,95,70                                                TAB2  45
   70 SCNT=SCNT+1.0                                                    TAB2  46
C        CALCULATE FREQUENCIES                                         TAB2  47
      TEMP1=UBO(1,1)-SINT(1)                                           TAB2  48
      DO 75 IY=1,INTY                                                  TAB2  49
      TEMP1=TEMP1+SINT(1)                                              TAB2  50
      IF(A(IJ)-TEMP1) 80,75,75                                         TAB2  51
   75 CONTINUE                                                         TAB2  52
      IY=INT1                                                          TAB2  53
   80 IYY=3*(IY-1)+1                                                   TAB2  54
      STAT1(IYY)=STAT1(IYY)+A(IJ)                                      TAB2  55
      IYY=IYY+1                                                        TAB2  56
      STAT1(IYY)=STAT1(IYY)+1.0                                        TAB2  57
      IYY=IYY+1                                                        TAB2  58
      STAT1(IYY)=STAT1(IYY)+A(IJ)*A(IJ)                                TAB2  59
      TEMP2=UBO(1,2)-SINT(2)                                           TAB2  60
      DO 85 IX=1,INTX                                                  TAB2  61
      TEMP2=TEMP2+SINT(2)                                              TAB2  62
      IF(A(IJX)-TEMP2) 90,85,85                                        TAB2  63
   85 CONTINUE                                                         TAB2  64
      IX=INT2                                                          TAB2  65
   90 IJF=INT1*(IX-1)+IY                                               TAB2  66
      FREQ(IJF)=FREQ(IJF)+1.0                                          TAB2  67
      IX=3*(IX-1)+1                                                    TAB2  68
      STAT2(IX)=STAT2(IX)+A(IJX)                                       TAB2  69
      IX=IX+1                                                          TAB2  70
      STAT2(IX)=STAT2(IX)+1.0                                          TAB2  71
      IX=IX+1                                                          TAB2  72
      STAT2(IX)=STAT2(IX)+A(IJX)*A(IJX)                                TAB2  73
   95 CONTINUE                                                         TAB2  74
C        CALCULATE PERCENT FREQUENCIES                                 TAB2  75
      DO 100 I=1,INTT                                                  TAB2  76
  100 PCT(I)=FREQ(I)*100.0/SCNT                                        TAB2  77
C        CALCULATE TOTALS, MEANS, STANDARD DEVIATIONS                  TAB2  78
      IXY=-1                                                           TAB2  79
      DO 120 I=1,INT1                                                  TAB2  80
      IXY=IXY+3                                                        TAB2  81
      ISD=IXY+1                                                        TAB2  82
      TEMP1=STAT1(IXY)                                                 TAB2  83
      SUM=STAT1(IXY-1)                                                 TAB2  84
      IF(TEMP1-1.0) 120,105,110                                        TAB2  85
  105 STAT1(ISD)=0.0                                                   TAB2  86
      GO TO 115                                                        TAB2  87
  110 STAT1(ISD)=SQRT(ABS((STAT1(ISD)-SUM*SUM/TEMP1)/(TEMP1-1.0)))     TAB2  88
  115 STAT1(IXY)=SUM/TEMP1                                             TAB2  89
  120 CONTINUE                                                         TAB2  90
      IXX=-1                                                           TAB2  91
      DO 140 I=1,INT2                                                  TAB2  92
      IXX=IXX+3                                                        TAB2  93
      ISD=IXX+1                                                        TAB2  94
      TEMP2=STAT2(IXX)                                                 TAB2  95
      SUM=STAT2(IXX-1)                                                 TAB2  96
      IF(TEMP2-1.0) 140,125,130                                        TAB2  97
  125 STAT2(ISD)=0.0                                                   TAB2  98
      GO TO 135                                                        TAB2  99
  130 STAT2(ISD)=SQRT(ABS((STAT2(ISD)-SUM*SUM/TEMP2)/(TEMP2-1.0)))     TAB2 100
  135 STAT2(IXX)=SUM/TEMP2                                             TAB2 101
  140 CONTINUE                                                         TAB2 102
      DO 150 I=1,3                                                     TAB2 M10
      DO 150 J=1,2                                                     TAB2 M11
  150 UBO(I,J)=WBO(I,J)                                                TAB2 M12
      RETURN                                                           TAB2 103
      END                                                              TAB2 104
```

SUBMX

Purpose:

Based on vector S derived from subroutine SUBST or ABSNT, this subroutine copies from a larger matrix of observation data a subset matrix of those observations which have satisfied certain condition. This subroutine is normally used prior to statistical analyses (e.g., multiple regression, factor analysis).

Usage:

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

Description of parameters:

- A - Input matrix of observations, NO by NV.
- D - Output matrix of observations, N by NV.
- S - Input vector of length NO containing the codes derived from subroutine SUBST or ABSNT.
- NO - Number of observations.
- NV - Number of variables.
- N - Output variable containing the number of non-zero codes in vector S.

Remarks:

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

Subroutines and function subprograms required:

None.

Method:

If S(I) contains a non-zero code, I-th observation is copied from the input matrix to the output matrix.

```
      SUBROUTINE SUBMX (A,D,S,NO,NV,N)                                  SUBMX  1
      DIMENSION A(1),D(1),S(1)                                          SUBMX  2
      L=0                                                               SUBMX  3
      LL=0                                                              SUBMX  4
      DO 20 J=1,NV                                                      SUBMX  5
      DO 15 I=1,NO                                                      SUBMX  6
      L=L+1                                                             SUBMX  7
      IF(S(I)) 15, 15, 10                                               SUBMX  8
   10 LL=LL+1                                                           SUBMX  9
      D(LL)=A(L)                                                        SUBMX 10
   15 CONTINUE                                                          SUBMX 11
   20 CONTINUE                                                          SUBMX 12
C        COUNT NON-ZERO CODES IN VECTOR S                               SUBMX 13
      N=0                                                               SUBMX 14
      DO 30 I=1,NO                                                      SUBMX 15
      IF(S(I)) 30, 30, 25                                               SUBMX 16
   25 N=N+1                                                             SUBMX 17
   30 CONTINUE                                                          SUBMX 18
      RETURN                                                            SUBMX 19
      END                                                               SUBMX 20
```

Statistics - Elementary

MOMEN

This subroutine computes four moments for grouped data $F_1, F_2, \ldots, F_n$ on equal class intervals. The number of class intervals is computed as follows:

$$n = (UBO_3 - UBO_1)/UBO_2 \qquad (1)$$

where UBO_1 = given lower bound

UBO_2 = given class interval

UBO_3 = given upper bound

and the total frequency as follows:

$$T = \sum_{i=1}^{n} F_i \qquad (2)$$

where F_i = frequency count in i-th interval.

Then, the following are computed:

First Moment (Mean):

$$ANS_1 = \frac{\sum_{i=1}^{n} F_i \left[UBO_1 + (i-0.5)\, UBO_2\right]}{T} \qquad (3)$$

j-th Moment (Variance):

$$ANS_j = \frac{\sum_{i=1}^{n} F_i \left[UBO_1 + (i-0.5)\, UBO_2 - ANS_1\right]^j}{T} \qquad (4)$$

$j = 2, 3, 4$

These moments are biased and not corrected for grouping

Subroutine MOMEN

Purpose:

To find the first four moments for grouped data on equal class intervals.

Usage:

CALL MOMEN (F, UBO, NOP, ANS)

Description of Parameters:

- F - Grouped data (frequencies). Given as a vector of length (UBO(3)-UBO(1))/UBO(2)
- UBO - 3 cell vector, UBO(1) is lower bound and UBO(3) upper bound on data. UBO(2) is 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:

Note that 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.

Subroutines and function subprograms required:

None.

Method:

Refer to M. G. Kendall, 'The Advanced Theory of Statistics', V.1, Hafner Publishing Company, 1958, Chapter 3.

```
      SUBROUTINE MOMEN (F,UBO,NOP,ANS)                                  MOMEN  1
      DIMENSION F(1),UBO(3),ANS(4)                                      MOMENM01
      DO 100 I=1,4                                                      MOMEN  3
  100 ANS(I)=0.0                                                        MOMEN  4
C         CALCULATE THE NUMBER OF CLASS INTERVALS                       MOMEN  5
      N=(UBO(3)-UBO(1))/UBO(2)+0.5                                      MOMENM02
C         CALCULATE TOTAL FREQUENCY                                     MOMEN  7
      T=0.0                                                             MOMEN  8
      DO 110 I=1,N                                                      MOMEN  9
  110 T=T+F(I)                                                          MOMEN 10
      IF(NOP-5) 130, 120, 115                                           MOMEN 11
  115 NOP=5                                                             MOMEN 12
  120 JUMP=1                                                            MOMEN 13
      GO TO 150                                                         MOMEN 14
  130 JUMP=2                                                            MOMEN 15
C         FIRST MOMENT                                                  MOMEN 16
  150 DO 160 I=1,N                                                      MOMEN 17
      FI=I                                                              MOMEN 18
  160 ANS(1)=ANS(1)+F(I)*(UBO(1)+(FI-0.5)*UBO(2))                       MOMEN 19
      ANS(1)=ANS(1)/T                                                   MOMEN 20
      GO TO (350,200,250,300,200), NOP                                  MOMEN 21
C         SECOND MOMENT                                                 MOMEN 22
  200 DO 210 I=1,N                                                      MOMEN 23
      FI=I                                                              MOMEN 24
  210 ANS(2)=ANS(2)+F(I)*(UBO(1)+(FI-0.5)*UBO(2)-ANS(1))**2             MOMEN 25
      ANS(2)=ANS(2)/T                                                   MOMEN 26
      GO TO (250,350), JUMP                                             MOMEN 27
C         THIRD MOMENT                                                  MOMEN 28
  250 DO 260 I=1,N                                                      MOMEN 29
      FI=I                                                              MOMEN 30
  260 ANS(3)=ANS(3)+F(I)*(UBO(1)+(FI-0.5)*UBO(2)-ANS(1))**3             MOMEN 31
      ANS(3)=ANS(3)/T                                                   MOMEN 32
      GO TO (300,350), JUMP                                             MOMEN 33
C         FOURTH MOMENT                                                 MOMEN 34
  300 DO 310 I=1,N                                                      MOMEN 35
      FI=I                                                              MOMEN 36
  310 ANS(4)=ANS(4)+F(I)*(UBO(1)+(FI-0.5)*UBO(2)-ANS(1))**4             MOMEN 37
      ANS(4)=ANS(4)/T                                                   MOMEN 38
  350 RETURN                                                            MOMEN 39
      END                                                               MOMEN 40
```

TTSTT

This subroutine computes certain t-statistics on the means of populations under various hypotheses.

The sample means of $A_1, A_2, \ldots, A_{NA}$ and $B_1, B_2, \ldots, B_{NB}$ are normally found by the following formulas:

$$\bar{A} = \frac{\sum_{i=1}^{NA} A_i}{NA}; \qquad \bar{B} = \frac{\sum_{i=1}^{NB} B_i}{NB} \tag{1}$$

and the corresponding sample variances by:

$$SA^2 = \frac{\sum_{i=1}^{NA} (A_i - \bar{A})^2}{NA - 1}; \qquad SB^2 = \frac{\sum_{i=1}^{NB} (B_i - \bar{B})^2}{NB - 1} \tag{2}$$

μ and σ^2 stand respectively for population mean and variance in the following hypotheses:

Hypothesis: $\mu_B = A$; A = a given value (Option 1):

Let $\bar{B}$ = estimate of μ_B and set NA = 1 (A is stored in location A).

The subroutine computes:

$$ANS = \frac{\bar{B} - A}{SB} \cdot \sqrt{NB} \quad \text{(t-statistic)} \tag{3}$$

$$NDF = NB - 1 \quad \text{(degrees of freedom)} \tag{4}$$

Hypothesis: $\mu_A = \mu_B$; $\sigma_A^2 = \sigma_B^2$ (Option 2):

The subroutine computes:

$$ANS = \frac{\bar{B} - \bar{A}}{S} \cdot \frac{1}{\sqrt{\frac{1}{NA} + \frac{1}{NB}}} \quad \text{(t-statistic)} \tag{5}$$

$$NDF = NA + NB - 2 \quad \text{(degrees of freedom)} \tag{6}$$

$$\text{where } S = \sqrt{\frac{(NA - 1)SA^2 + (NB - 1)SB^2}{NA + NB - 2}} \tag{7}$$

Hypothesis: $\mu_A = \mu_B \left(\sigma_A^2 \neq \sigma_B^2\right)$ (Option 3):

The subroutine computes:

$$ANS = \frac{\bar{B} - \bar{A}}{\sqrt{\frac{SA^2}{NA} + \frac{SB^2}{NB}}} \quad \text{(t-statistic)} \tag{8}$$

$$NDF = \frac{\left(\frac{SA^2}{NA} + \frac{SB^2}{NB}\right)^2}{\left(\frac{SA^2}{NA}\right)^2 / (NA+1) + \left(\frac{SB^2}{NB}\right)^2 / (NB+1)} - 2 \qquad (9)$$

(degrees of freedom)

Note: The program returns a rounded NDF, not a truncated NDF.

Hypothesis: $\mu_A = \mu_B$ (no assumption on σ^2) (Option 4):

The subroutine computes:

$$ANS = \frac{\bar{D}}{SD} \cdot \sqrt{NB} \qquad \text{(t-statistic)} \qquad (10)$$

$$NDF = NB - 1 \qquad \text{(degrees of freedom)} \qquad (11)$$

where $\bar{D} = \bar{B} - \bar{A}$ (12)

$$SD = \sqrt{\frac{\sum_{i=1}^{NB} (B_i - A_i - \bar{D})^2}{NB - 1}} \qquad (13)$$

NA = NB

Subroutine TTSTT

Purpose:

To find certain T-statistics on the means of populations.

Usage:

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

Description of parameters:

- 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:
  - NOP=1--- That population mean of B = given value A. (Set NA=1.)
  - NOP=2--- That population mean of B = population mean of A, given that the variance of B = the variance of A.
  - NOP=3--- That population mean of B = population mean of A, given that the variance of B is not equal to the variance of A.
  - NOP=4--- That population mean of B = population mean of A, given no information about variances of A and B. (Set NA=NB.)
- NDF - Output variable containing degrees of freedom associated with 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 will not be calculated. NDF and ANS will be set to zero.

Subroutines and function subprograms required:

None.

Method:

Refer to Ostle, Bernard, 'Statistics in Research', Iowa State College Press, 1954, Chapter 5.

```
      SUBROUTINE TTSTT (A,NA,B,NB,NOP,NDF,ANS)                          TTSTT  1
      DIMENSION A(1),B(1)                                               TTSTT  2
C        INITIALIZATION                                                 TTSTT  3
      NDF=0                                                             TTSTT  4
      ANS=0.0                                                           TTSTT  5
C        CALCULATE THE MEAN OF A                                        TTSTT  6
      AMEAN=0.0                                                         TTSTT  7
      DO 110 I=1,NA                                                     TTSTT  8
  110 AMEAN=AMEAN+A(I)                                                  TTSTT  9
      FNA=NA                                                            TTSTT 10
      AMEAN=AMEAN/FNA                                                   TTSTT 11
C        CALCULATE THE MEAN OF B                                        TTSTT 12
  115 BMEAN=0.0                                                         TTSTT 13
      DO 120 I=1,NB                                                     TTSTT 14
  120 BMEAN=BMEAN+B(I)                                                  TTSTT 15
      FNB=NB                                                            TTSTT 16
      BMEAN=BMEAN/FNB                                                   TTSTT 17
      IF(NOP-4) 122, 180, 200                                           TTSTT 18
  122 IF(NOP-1) 200, 135, 125                                           TTSTT 19
C        CALCULATE THE VARIANCE OF A                                    TTSTT 20
  125 SA2=0.0                                                           TTSTT 21
      DO 130 I=1,NA                                                     TTSTT 22
  130 SA2=SA2+(A(I)-AMEAN)**2                                           TTSTT 23
      SA2=SA2/(FNA-1.0)                                                 TTSTT 24
C        CALCULATE THE VARIANCE OF B                                    TTSTT 25
  135 SB2=0.0                                                           TTSTT 26
      DO 140 I=1,NB                                                     TTSTT 27
  140 SB2=SB2+(B(I)-BMEAN)**2                                           TTSTT 28
      SB2=SB2/(FNB-1.0)                                                 TTSTT 29
      GO TO (150,160,170), NOP                                          TTSTT 30
C        OPTION 1                                                       TTSTT 31
  150 ANS=(BMEAN-AMEAN)/SQRT(SB2)*SQRT(FNB)                             TTSTT 32
      NDF=NB-1                                                          TTSTT 33
      GO TO 200                                                         TTSTT 34
C        OPTION 2                                                       TTSTT 35
  160 NDF=NA+NB-2                                                       TTSTT 36
      FNDF=NDF                                                          TTSTT 37
      S=SQRT(((FNA-1.0)*SA2+(FNB-1.0)*SB2)/FNDF)                        TTSTT 38
      ANS=(BMEAN-AMEAN)/S*(1.0/SQRT(1.0/FNA+1.0/FNB))                   TTSTT 39
      GO TO 200                                                         TTSTT 40
C        OPTION 3                                                       TTSTT 41
  170 ANS=(BMEAN-AMEAN)/SQRT(SA2/FNA+SB2/FNB)                           TTSTT 42
      A1=(SA2/FNA+SB2/FNB)**2                                           TTSTT 43
      A2=(SA2/FNA)**2/(FNA+1.0)+(SB2/FNB)**2/(FNB+1.0)                  TTSTT 44
      NDF=A1/A2-2.0+0.5                                                 TTSTT 45
      GO TO 200                                                         TTSTT 46
C        OPTION 4                                                       TTSTT 47
  180 SD=0.0                                                            TTSTT 48
      D=BMEAN-AMEAN                                                     TTSTT 49
      DO 190 I=1,NB                                                     TTSTT 50
  190 SD=SD+(B(I)-A(I)-D)**2                                            TTSTT 51
      SD=SQRT(SD/(FNB-1.0))                                             TTSTT 52
      ANS=(D/SD)*SQRT(FNB)                                              TTSTT 53
      NDF=NB-1                                                          TTSTT 54
  200 RETURN                                                            TTSTT 55
      END                                                               TTSTT 56
```

CORRE

This subroutine calculates means, standard deviations, sums of cross-products of deviations from means, and product moment correlation coefficients from input data X_{ij}, where i = 1, 2, ..., n implies observations and j = 1, 2, ..., m implies variables.

The following equations are used to calculate these statistics:

Sums of cross-products of deviations:

$$S_{jk} = \sum_{i=1}^{n} \left(X_{ij} - T_j\right)\left(X_{ik} - T_k\right) - \frac{\sum_{i=1}^{n} \left(X_{ij} - T_j\right) \sum_{i=1}^{n} \left(X_{ik} - T_k\right)}{n} \quad (1)$$

where j = 1, 2, ..., m; k = 1, 2, ..., m

$$T_j = \frac{\sum_{i=1}^{m} X_{ij}}{m} \quad (2)$$

(These temporary means T_j are subtracted from the data in equation (1) to obtain computational accuracy.)

Means: $$\bar{X}_j = \frac{\sum_{i=1}^{n} X_{ij}}{n} \quad (3)$$

where j = 1, 2, ..., m

Correlation coefficients:

$$r_{jk} = \frac{S_{jk}}{\sqrt{S_{jj}}\ \sqrt{S_{kk}}} \quad (4)$$

where j = 1, 2, ..., m; k = 1, 2, ..., m

Standard deviations:

$$s_j = \frac{\sqrt{S_{jj}}}{\sqrt{n-1}} \quad (5)$$

where j = 1, 2, ..., m

Purpose:

Compute means, standard deviations, sums of cross-products of deviations, and correlation coefficients.

Usage:

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

Description of parameters:

- N - Number of observations.
- M - Number of variables.
- IO - Option code for input data.
  - 0 If data are to be read in from input device in the special subroutine named data. (See "subroutines and function subprograms required" below.)
  - 1 If all data are already in core.
- 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.
- 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.

Remarks:

None.

Subroutines and function subprograms required:

- DATA(M,D) - This subroutine must be provided by the user.
  - (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 exist in job deck. If user has not supplied a subroutine named DATA, the following is suggested.

```
      SUBROUTINE DATA
      RETURN
      END
```

Method:

Product-moment correlation coefficients are computed.

```
      SUBROUTINE CORRE (N,M,IO,X,XBAR,STD,RX,R,B,D,T)                    CORRE  1
      DIMENSION X(1),XBAR(1),STD(1),RX(1),R(1),B(1),D(1),T(1)            CORRE  2
C        INITIALIZATION                                                  CORRE  3
      DO 100 J=1,M                                                       CORRE  4
      B(J)=0.0                                                           CORRE  5
  100 T(J)=0.0                                                           CORRE  6
      K=(M*M+M)/2                                                        CORRE  7
      DO 102 I=1,K                                                       CORRE  8
  102 R(I)=0.0                                                           CORRE  9
      FN=N                                                               CORRE 10
      L=0                                                                CORRE 11
      IF(IO) 105, 127, 105                                               CORRE 12
C        DATA ARE ALREADY IN CORE                                        CORRE 13
  105 DO 108 J=1,M                                                       CORRE 14
      DO 107 I=1,N                                                       CORRE 15
      L=L+1                                                              CORRE 16
  107 T(J)=T(J)+X(L)                                                     CORRE 17
      XBAR(J)=T(J)                                                       CORRE 18
  108 T(J)=T(J)/FN                                                       CORRE 19
      DO 115 I=1,N                                                       CORRE 20
      JK=0                                                               CORRE 21
      L=I-N                                                              CORRE 22
      DO 110 J=1,M                                                       CORRE 23
      L=L+N                                                              CORRE 24
      D(J)=X(L)-T(J)                                                     CORRE 25
  110 B(J)=B(J)+D(J)                                                     CORRE 26
      DO 115 J=1,M                                                       CORRE 27
      DO 115 K=1,J                                                       CORRE 28
      JK=JK+1                                                            CORRE 29
  115 R(JK)=R(JK)+D(J)*D(K)                                              CORRE 30
      GO TO 205                                                          CORRE 31
C        READ OBSERVATIONS AND CALCULATE TEMPORARY                       CORRE 32
C        MEANS FROM THESE DATA IN T(J)                                   CORRE 33
  127 IF(N-M) 130, 130, 135                                              CORRE 34
  130 KK=N                                                               CORRE 35
      GO TO 137                                                          CORRE 36
  135 KK=M                                                               CORRE 37
  137 DO 140 I=1,KK                                                      CORRE 38
      CALL DATA (M,D)                                                    CORRE 39
      DO 140 J=1,M                                                       CORRE 40
      T(J)=T(J)+D(J)                                                     CORRE 41
      L=L+1                                                              CORRE 42
  140 RX(L)=D(J)                                                         CORRE 43
      FKK=KK                                                             CORRE 44
      DO 150 J=1,M                                                       CORRE 45
      XBAR(J)=T(J)                                                       CORRE 46
  150 T(J)=T(J)/FKK                                                      CORRE 47
C        CALCULATE SUMS OF CROSS-PRODUCTS OF DEVIATIONS                  CORRE 48
C        FROM TEMPORARY MEANS FOR M OBSERVATIONS                         CORRE 49
      L=0                                                                CORRE 50
      DO 180 I=1,KK                                                      CORREM06
      JK=0                                                               CORRE 52
      DO 170 J=1,M                                                       CORRE 53
      L=L+1                                                              CORRE 54
  170 D(J)=RX(L)-T(J)                                                    CORRE 55
      DO 180 J=1,M                                                       CORRE 56
      B(J)=B(J)+D(J)                                                     CORRE 57
      DO 180 K=1,J                                                       CORRE 58
      JK=JK+1                                                            CORRE 59
  180 R(JK)=R(JK)+D(J)*D(K)                                              CORRE 60
      IF(N-KK) 205, 205, 185                                             CORRE 61
C        READ THE REST OF OBSERVATIONS ONE AT A TIME, SUM                CORRE 62
C        THE OBSERVATION, AND CALCULATE SUMS OF CROSS-                   CORRE 63
C        PRODUCTS OF DEVIATIONS FROM TEMPORARY MEANS                     CORRE 64
  185 KK=N-KK                                                            CORRE 65
      DO 200 I=1,KK                                                      CORRE 66
      JK=0                                                               CORRE 67
      CALL DATA (M,D)                                                    CORRE 68
      DO 190 J=1,M                                                       CORRE 69
      XBAR(J)=XBAR(J)+D(J)                                               CORRE 70
      D(J)=D(J)-T(J)                                                     CORRE 71
  190 B(J)=B(J)+D(J)                                                     CORRE 72
      DO 200 J=1,M                                                       CORRE 73
      DO 200 K=1,J                                                       CORRE 74
      JK=JK+1                                                            CORRE 75
  200 R(JK)=R(JK)+D(J)*D(K)                                              CORRE 76
C        CALCULATE MEANS                                                 CORRE 77
  205 JK=0                                                               CORRE 78
      DO 210 J=1,M                                                       CORRE 79
      XBAR(J)=XBAR(J)/FN                                                 CORRE 80
C        ADJUST SUMS OF CROSS-PRODUCTS OF DEVIATIONS                     CORRE 81
C        FROM TEMPORARY MEANS                                            CORRE 82
      DO 210 K=1,J                                                       CORRE 83
      JK=JK+1                                                            CORRE 84
  210 R(JK)=R(JK)-B(J)*B(K)/FN                                           CORRE 85
C        CALCULATE CORRELATION COEFFICIENTS                              CORRE 86
      JK=0                                                               CORRE 87
      DO 220 J=1,M                                                       CORRE 88
      JK=JK+J                                                            CORRE 89
  220 STD(J)= SQRT( ABS(R(JK)))                                          CORRE 90
      DO 230 J=1,M                                                       CORRE 91
      DO 230 K=J,M                                                       CORRE 92
      JK=J+(K*K-K)/2                                                     CORRE 93
      L=M*(J-1)+K                                                        CORRE 94
      RX(L)=R(JK)                                                        CORRE 95
      L=M*(K-1)+J                                                        CORRE 96
      RX(L)=R(JK)                                                        CORRE 97
      IF(STD(J)*STD(K))225,222,225                                       CORREM01
  222 R(JK)=0.0                                                          CORREM02
      GO TO 230                                                          CORREM03
  225 R(JK)=R(JK)/(STD(J)*STD(K))                                        CORREM04
  230 CONTINUE                                                           CORREM05
C        CALCULATE STANDARD DEVIATIONS                                   CORRE 99
      FN=SQRT(FN-1.0)                                                    CORRE100
      DO 240 J=1,M                                                       CORRE101
                                                                         CORRE102
  240 STD(J)=STD(J)/FN                                                   CORRE103
C        COPY THE DIAGONAL OF THE MATRIX OF SUMS OF CROSS-PRODUCTS OF    CORRE104
C        DEVIATIONS FROM MEANS.                                          CORRE105
      L=-M                                                               CORRE106
      DO 250 I=1,M                                                       CORRE107
      L=L+M+1                                                            CORRE108
  250 B(I)=RX(L)                                                         CORRE109
      RETURN                                                             CORRE110
      END
```

Statistics - Multiple Linear Regression

In the Scientific Subroutine Package, multiple linear regression is normally performed by calling four subroutines in sequence.

1. CORRE - to find means, standard deviations, and correlation matrix
2. ORDER - to choose a dependent variable and a subset of independent variables from a larger set of variables
3. MINV - to invert the correlation matrix of the subset selected by ORDER
4. MULTR - to compute the regression coefficients, b_0, b_1, b_2, ..., b_m, and various confidence measures

The subroutine CORRE works in either of two ways: (1) it expects all observations in core, or (2) it triggers a user-provided input subroutine, DATA, to read one observation at a time into a work area. In either case, the user must provide a subroutine named DATA (see "Subroutines Required" in the description of subroutine CORRE).

ORDER

Purpose:

Construct from a larger matrix of correlation coefficients a subset matrix of intercorrelations among independent variables and a vector of intercorrelations of independent variables with dependent variable. This subroutine is normally used in the performance of multiple and polynomial regression analyses.

Usage:

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

Description of parameters:

M - Number of variables and order of matrix R.

R - Input matrix containing correlation coefficients. This subroutine expects only 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.

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 K+1 position.

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.

Remarks:

None.

Subroutines and function subprograms required:

None.

Method:

From the subscript numbers of the variables to be included in the forthcoming regression, the subroutine constructs the matrix RX and the vector RY.

```
      SUBROUTINE ORDER (M,R,NDEP,K,ISAVE,RX,RY)                         ORDER  1
      DIMENSION R(1),ISAVE(1),RX(1),RY(1)                               ORDER  2
C        COPY INTERCORRELATIONS OF INDEPENDENT VARIABLES                 ORDER  3
C        WITH DEPENDENT VARIABLE                                         ORDER  4
      MM=0                                                              ORDER  5
      DO 130 J=1,K                                                      ORDER  6
      L2=ISAVE(J)                                                       ORDER  7
      IF(NDEP-L2) 122, 123, 123                                         ORDER  8
  122 L=NDEP*(L2*L2-L2)/2                                               ORDER  9
      GO TO 125                                                         ORDER 10
  123 L=L2+(NDEP*NDEP-NDEP)/2                                           ORDER 11
  125 RY(J)=R(L)                                                        ORDER 12
C        COPY A SUBSET MATRIX OF INTERCORRELATIONS AMONG                 ORDER 13
C        INDEPENDENT VARIABLES                                           ORDER 14
      DO 130 I=1,K                                                      ORDER 15
      LI=ISAVE(I)                                                       ORDER 16
      IF(LI-L2) 127, 129, 128                                           ORDER 17
  127 L=LI+(L2*L2-L2)/2                                                 ORDER 18
      GO TO 129                                                         ORDER 19
  128 L=L2+(LI*LI-LI)/2                                                 ORDER 20
  129 MM=MM+1                                                           ORDER 21
  130 RX(MM)=R(L)                                                       ORDER 22
C        PLACE THE SUBSCRIPT NUMBER OF THE DEPENDENT                    ORDER 23
C        VARIABLE IN ISAVE(K+1)                                         ORDER 24
      ISAVE(K+1)=NDEP                                                   ORDER 25
      RETURN                                                            ORDER 26
      END                                                               ORDER 27
```

MULTR

This subroutine performs a multiple regression analysis for a dependent variable and a set of independent variables.

Beta weights are calculated using the following equation:

$$\beta_j = \sum_{i=1}^{k} r_{iy} \cdot r_{ij}^{-1} \quad (1)$$

where r_{iy} = intercorrelation of i^{th} independent variable with dependent variable

r_{ij}^{-1} = the inverse of intercorrelation r_{ij}

i, j = 1, 2, ..., k imply independent variables

r_{iy} and r_{ij}^{-1} are input to this subroutine.

Then, the regression coefficients are calculated as follows:

$$b_j = \beta_j \cdot \frac{s_y}{s_j} \quad (2)$$

where s_y = standard deviation of dependent variable

s_j = standard deviation of j^{th} independent variable

j = 1, 2, ..., k

s_y and s_j are input to this subroutine.

The intercept is found by the following equation:

$$b_0 = \bar{Y} - \sum_{j=1}^{k} b_j \cdot \bar{X}_j \quad (3)$$

where $\bar{Y}$ = mean of dependent variable

$\bar{X}_j$ = mean of j^{th} independent variable

$\bar{Y}$ and $\bar{X}_j$ are input to this subroutine.

Multiple correlation coefficient, R, is found first by calculating the coefficient of determination by the following equation:

$$R^2 = \sum_{i=1}^{k} \beta_i r_{iy} \quad (4)$$

and taking the square root of R^2:

$$R = \sqrt{R^2} \quad (5)$$

The sum of squares attributable to the regression is found by:

$$SSAR = R^2 \cdot D_{yy} \quad (6)$$

where D_{yy} = sum of squares of deviations from mean for dependent variable

D_{yy} is input to this subroutine.

The sum of squares of deviations from the regression is obtained by:

$$SSDR = D_{yy} - SSAR \quad (7)$$

Then, the F-value for the analysis of variance is calculated as follows:

$$F = \frac{SSAR/k}{SSDR/(n-k-1)} = \frac{SSAR(n-k-1)}{SSDR(k)} \quad (8)$$

Certain other statistics are calculated as follows:

Variance and standard error of estimate:

$$S^2_{y.12\ldots k} = \frac{SSDR}{n-k-1} \quad (9)$$

where n = number of observations

$$S_{y.12\ldots k} = \sqrt{S^2_{y.12\ldots k}} \quad (10)$$

Standard deviations of regression coefficients:

$$S_{b_j} = \sqrt{\frac{r_{jj}^{-1}}{D_{jj}} \cdot S^2_{y.12\ldots k}} \quad (11)$$

where D_{jj} = sum of squares of deviations from mean for j^{th} independent variable. D_{jj} is input to this subroutine.

j = 1, 2, ..., k

Computed t:

$$t_j = \frac{b_j}{S_{b_j}} \quad (12)$$

j = 1, 2, ..., k

Subroutine MULTR

Purpose:

Perform a multiple linear regression analysis for a dependent variable and a set of independent variables. This subroutine is normally used in the performance of multiple and polynomial regression analyses.

Usage:

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

Description of parameters:

- N - Number of observations.
- K - Number of independent variables in this regression.
- XBAR - Input vector of length M containing means of all variables. M is number of variables in observations.
- 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 variables.
- 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 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, 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 attributable to regression (SSAR)
  - ANS(5) Degrees of freedom associated with SSAR
  - ANS(6) Mean square of SSAR
  - ANS(7) Sum of squares of deviations from regression (SSDR)
  - ANS(8) Degrees of freedom associated with SSDR
  - ANS(9) Mean square of SSDR
  - ANS(10) F-value

Remarks:

N must be greater than K+1.

Subroutines and function subprograms required:

None.

Method:

The Gauss-Jordan method is used in the solution of the normal equations. Refer to W. W. Cooley and P. R. Lohnes, 'Multivariate Procedures for the Behavioral Sciences', John Wiley and Sons, 1962, Chapter 3, and B. Ostle, 'Statistics in Research', The Iowa State College Press, 1954, Chapter 8.

```
      SUBROUTINE MULTR (N,K,XBAR,STD,D,RX,RY,ISAVE,B,SB,T,ANS)          MULTR  1
      DIMENSION XBAR(1),STD(1),D(1),RX(1),RY(1),ISAVE(1),B(1),SB(1),    MULTR  2
     1          T(1),ANS(10)                                            MULTRM01
      MM=K+1                                                            MULTR  4
C        BETA WEIGHTS                                                   MULTR  5
      DO 100 J=1,K                                                      MULTR  6
  100 B(J)=0.0                                                          MULTR  7
      DO 110 J=1,K                                                      MULTR  8
      LI=K*(J-1)                                                        MULTR  9
      DO 110 I=1,K                                                      MULTR 10
      L=LI+I                                                            MULTR 11
  110 B(J)=B(J)+RY(I)*RX(L)                                             MULTR 12
      RM=0.0                                                            MULTR 13
      BO=0.0                                                            MULTR 14
      LI=ISAVE(MM)                                                      MULTR 15
C        COEFFICIENT OF DETERMINATION                                   MULTR 16
      DO 120 I=1,K                                                      MULTR 17
      RM=RM+B(I)*RY(I)                                                  MULTR 18
C        REGRESSION COEFFICIENTS                                        MULTR 19
      L=ISAVE(I)                                                        MULTR 20
      B(I)=B(I)*(STD(LI)/STD(L))                                        MULTR 21
C        INTERCEPT                                                      MULTR 22
  120 BO=BO+B(I)*XBAR(L)                                                MULTR 23
      BO=XBAR(LI)-BO                                                    MULTR 24
C        SUM OF SQUARES ATTRIBUTABLE TO REGRESSION                      MULTR 25
      SSAR=RM*D(LI)                                                     MULTR 26
C        MULTIPLE CORRELATION COEFFICIENT                               MULTR 27
  122 RM= SQRT( ANS(RM) )                                               MULTR 28
C        SUM OF SQUARES OF DEVIATIONS FROM REGRESSION                   MULTR 29
      SSDR=D(LI)-SSAR                                                   MULTR 30
C        VARIANCE OF ESTIMATE                                           MULTR 31
      FN=N-K-1                                                          MULTR 32
      SY=SSDR/FN                                                        MULTR 33
C        STANDARD DEVIATIONS OF REGRESSION COEFFICIENTS                 MULTR 34
      DO 130 J=1,K                                                      MULTR 35
      LI=K*(J-1)+J                                                      MULTR 36
      L=ISAVE(J)                                                        MULTR 37
  125 SB(J)= SQRT( ABS((RX(LI)/D(LI))*SY)  )                            MULTR 38
C        COMPUTED T-VALUES                                              MULTR 39
  130 T(J)=B(J)/SB(J)                                                   MULTR 40
C        STANDARD ERROR OF ESTIMATE                                     MULTR 41
  135 SY= SQRT( ABS(SY) )                                               MULTR 42
C        F VALUE                                                        MULTR 43
      FK=K                                                              MULTR 44
      SSARM=SSAR/FK                                                     MULTR 45
      SSDRM=SSDR/FN                                                     MULTR 46
      F=SSARM/SSDRM                                                     MULTR 47
      ANS(1)=BO                                                         MULTR 48
      ANS(2)=RM                                                         MULTR 49
      ANS(3)=SY                                                         MULTR 50
      ANS(4)=SSAR                                                       MULTR 51
      ANS(5)=FK                                                         MULTR 52
      ANS(6)=SSARM                                                      MULTR 53
      ANS(7)=SSDR                                                       MULTR 54
      ANS(8)=FN                                                         MULTR 55
      ANS(9)=SSDRM                                                      MULTR 56
      ANS(10)=F                                                         MULTR 57
      RETURN                                                            MULTR 58
      END                                                               MULTR 59
```

Statistics - Polynomial Regression

Polynomial regression is a statistical technique for finding the coefficients, b_0, b_1, b_2, ..., b_m, in the functional relationship of the form:

$$y = b_0 + b_1 x + b_2 x^2 + \ldots + b_m x^m$$

between a dependent variable y and a single independent variable x.

In the Scientific Subroutine Package, polynomial regression is normally performed by calling the following four subroutines in sequence:

1. GDATA - to generate the powers of the independent variable and find means, standard deviations, and correlation matrix
2. ORDER - to choose a dependent variable and subset of independent variables from a larger set of variables
3. MINV - to invert the correlation coefficient matrix
4. MULTR - to compute the regression coefficients, b_0, b_1, b_2, ..., b_m, and various confidence measures

The special subroutine PLOT may be used to plot Y values and Y estimates.

GDATA

This subroutine generates independent variables up to the m^{th} power (the highest degree polynomial specified) and calculates means, standard deviations, sums of cross-products of deviations from means, and product moment correlation coefficients.

X_{i1} denotes the i^{th} case of the independent variable;

X_{ip} denotes the i^{th} case of the dependent variable,

where $i = 1, 2, \ldots, n$

n - number of cases (observations)

$p = m + 1$

m = highest degree polynomial specified

The subroutine GDATA generates powers of the independent variable as follows:

$$\begin{aligned} X_{i2} &= X_{i1} \cdot X_{i1} \\ X_{i3} &= X_{i2} \cdot X_{i1} \\ X_{i4} &= X_{i3} \cdot X_{i1} \\ &\cdot \\ &\cdot \\ &\cdot \\ X_{im} &= X_{i,m-1} \cdot X_{i1} \end{aligned} \qquad (1)$$

where i and m are as defined as above.

Then, the following are calculated:

Means:

$$\bar{X}_j = \frac{\sum_{i=1}^{n} X_{ij}}{n} \qquad (2)$$

where $j = 1, 2, \ldots, p$

Sums of cross-products of deviations from means:

$$D_{jk} = \sum_{i=1}^{n}\left(X_{ij} - \bar{X}_j\right)\left(X_{ik} - \bar{X}_k\right) - \frac{\sum_{i=1}^{n}\left(X_{ij} - \bar{X}_j\right)\sum_{i=1}^{n}\left(X_{ik} - \bar{X}_k\right)}{n} \quad (3)$$

where j = 1, 2, ..., p; k = 1, 2, ..., p.

Correlation coefficients:

$$r_{ij} = \frac{D_{ij}}{\sqrt{D_{ii}}\ \sqrt{D_{jj}}} \quad (4)$$

where i = 1, 2, ..., p; j = 1, 2, ..., p.

Standard deviations:

$$s_j = \frac{\sqrt{D_{jj}}}{\sqrt{n-1}} \quad (5)$$

where j = 1, 2, ..., p

Subroutine GDATA

Purpose:

Generate independent variables up to the M^{th} power (the highest degree polynomial specified) and compute means, standard deviations, and correlation coefficients. This subroutine is normally called before subroutines ORDER, MINV and MULTR in the performance of a polynomial regression.

Usage:

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

Description of parameters:

- N - Number of observations.
- M - The highest degree polynomial to be fitted.
- X - Input matrix (N by M+1). When the subroutine is called, data for the independent variable are stored in the first column of matrix X, and data for the dependent variable are 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.

Subroutines and function subprograms required:

None.

Method:

Refer to B. Ostle, 'Statistics in Research', The Iowa State College Press, 1954, Chapter 6.

```
      SUBROUTINE GDATA (N,M,X,XBAR,STD,D,SUMSQ)                         GDATA  1
      DIMENSION X(1),XBAR(1),STD(1),D(1),SUMSQ(1)                       GDATA  2
C        GENERATE INDEPENDENT VARIABLES                                 GDATA  3
      IF (M-1 ) 105,105,90                                              GDATA  4
   90 L1=0                                                              GDATA  5
      DO 100 I=2,M                                                      GDATA  6
      L1=L1+N                                                           GDATA  7
      DO 100 J=1,N                                                      GDATA  8
      L=L1+J                                                            GDATA  9
      K=L-N                                                             GDATA 10
  100 X(L)=X(K)*X(J)                                                    GDATA 11
C        CALCULATE MEANS                                                GDATA 12
  105 MM=M+1                                                            GDATA 13
      DF=N                                                              GDATA 14
      L=0                                                               GDATA 15
      DO 115 I=1,MM                                                     GDATA 16
      XBAR(I)=0.0                                                       GDATA 17
      DO 110 J=1,N                                                      GDATA 18
      L=L+1                                                             GDATA 19
  110 XBAR(I)=XBAR(I)+X(L)                                              GDATA 20
  115 XBAR(I)=XBAR(I)/DF                                                GDATA 21
      DO 130 I=1,MM                                                     GDATA 22
  130 STD(I)=0.0                                                        GDATA 23
C        CALCULATE SUMS OF CROSS-PRODUCTS OF DEVIATIONS                 GDATA 24
      L=((MM+1)*MM)/2                                                   GDATA 25
      DO 150 I=1,L                                                      GDATA 26
  150 D(I)=0.0                                                          GDATA 27
      DO 170 K=1,N                                                      GDATA 28
      L=0                                                               GDATA 29
      DO 170 J=1,MM                                                     GDATA 30
      L2=N*(J-1)+K                                                      GDATA 31
      T2=X(L2)-XBAR(J)                                                  GDATA 32
      STD(J)=STD(J)+T2                                                  GDATA 33
      DO 170 I=1,J                                                      GDATA 34
      L1=N*(I-1)+K                                                      GDATA 35
      T1=X(L1)-XBAR(I)                                                  GDATA 36
      L=L+1                                                             GDATA 37
  170 D(L)=D(L)+T1*T2                                                   GDATA 38
      L=0                                                               GDATA 39
      DO 175 J=1,MM                                                     GDATA 40
      DO 175 I=1,J                                                      GDATA 41
      L=L+1                                                             GDATA 42
  175 D(L)=D(L)-STD(I)*STD(J)/DF                                        GDATA 43
      L=0                                                               GDATA 44
      DO 180 I=1,MM                                                     GDATA 45
      L=L+I                                                             GDATA 46
      SUMSQ(I)=D(L)                                                     GDATA 47
  180 STD(I)= SQRT( ABS(D(L)))                                          GDATA 48
C        CALCULATE CORRELATION COEFFICIENTS                             GDATA 49
      L=0                                                               GDATA 50
      DO 190 J=1,MM                                                     GDATA 51
      DO 190 I=1,J                                                      GDATA 52
      L=L+1                                                             GDATA 53
  190 D(L)=D(L)/(STD(I)*STD(J))                                         GDATA 54
C        CALCULATE STANDARD DEVIATIONS                                  GDATA 55
      DF=SQRT(DF-1.0)                                                   GDATA 56
      DO 200 I=1,MM                                                     GDATA 57
  200 STD(I)=STD(I)/DF                                                  GDATA 58
      RETURN                                                            GDATA 59
      END                                                               GDATA 60
```

Statistics - Canonical Correlation

In the Scientific Subroutine Package, canonical correlation analysis is normally performed by calling the following five subroutines:

1. CORRE - to compute means, standard deviations, and correlation matrix
2. MINV - to invert a part of the correlation matrix
3. EIGEN - to compute eigenvalues and eigenvectors
4. NROOT - to compute eigenvalues and eigenvectors of real nonsymmetric matrix of the form $B^{-1}A$
5. CANOR - to compute canonical correlations and coefficients

The subroutine CORRE works in either of two ways: (1) it expects all observations in core, or (2) it triggers a user-provided input subroutine, DATA, to read one observation at a time into a work area. In either case, the user must provide a subroutine named DATA (see "Subroutines Required" in the description of subroutine CORRE).

CANOR

This subroutine performs a canonical correlation analysis between two sets of variables.

The matrix of intercorrelations, R, is partitioned into four submatrices:

$$R = \left[\begin{array}{c|c} R_{11} & R_{12} \\ \hline R_{21} & R_{22} \end{array}\right] \quad (1)$$

R_{11} = intercorrelations among p variables in the first set (that is, left-hand variables)

R_{12} = intercorrelations between the variables in the first and second sets

R_{21} = the transpose of R_{12}

R_{22} = intercorrelations among q variables in the second set (that is, right-hand variables)

The equation:

$$\left| R_{22}^{-1} R_{21} R_{11}^{-1} R_{12} - \lambda I \right| = 0 \quad (2)$$

is then solved for all values of λ, eigenvalues, in the following matrix operation:

$$T = R_{11}^{-1} R_{12} \quad (3)$$

$$A = R_{21} T \quad (4)$$

The subroutine NROOT calculates eigenvalues (λ_i) with associated eigenvectors of $R_{22}^{-1} A$, where i = 1, 2, ..., q.

For each subscript i = 1, 2, ..., q, the following statistics are calculated:

Canonical correlation:

$$CANR = \sqrt{\lambda_i} \quad (5)$$

where λ_i = i^{th} eigenvalue

Chi-square:

$$\chi^2 = - \left[n - 0.5 (p + q + 1)\right] \log_e \Lambda \quad (6)$$

where n = number of observations

$$\Lambda = \prod_{j=1}^{q} (1 - \lambda_j);$$

Degrees of freedom for χ^2:

$$DF = \left[p-(i-1)\right] \left[q-(i-1)\right]; \quad (7)$$

i^{th} set of right-hand coefficients:

$$b_k = v_{ki} \quad (8)$$

where v_{ki} = eigenvector associated with λ_i

$k = 1, 2, \ldots, q;$

i^{th} set of left-hand coefficients:

$$a_j = \frac{\sum_{k=1}^{q} t_{jk} b_k}{CANR} \quad (9)$$

where $\{t_{jk}\} = T = R_{11}^{-1} R_{12}$

$j = 1, 2, \ldots, p$

Subroutine CANOR

Purpose:

Compute the canonical correlations between two sets of variables. CANOR is normally preceded by a call to subroutine CORRE.

Usage:

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

Description of parameters:

- 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.)
- 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 columnwise.
- COEFL - Output matrix (MP by MQ) containing MQ sets of left hand coefficients columnwise.
- 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 which are greater than zero.

Subroutines and function subprograms required:

MINV

NROOT (which, in turn, calls the subroutine EIGEN.)

Method:

Refer to W. W. Cooley and P. R. Lohnes, 'Multivariate Procedures for the Behavioral Sciences', John Wiley and Sons, 1962, Chapter 3.

```
      SUBROUTINE CANOR (N,MP,MQ,RR,ROOTS,WLAM,CANR,CHISQ,NDF,COEFR,     CANOR  1
     1                  COEFL,R)                                        CANOR  2
      DIMENSION RR(1),ROOTS(1),WLAM(1),CANR(1),CHISQ(1),NDF(1),COEFR(1),CANOR  3
     1          COEFL(1),R(1)                                           CANOR  4
C        PARTITION INTERCORRELATIONS AMONG LEFT HAND VARIABLES, BETWEEN CANOR  5
C        LEFT AND RIGHT HAND VARIABLES, AND AMONG RIGHT HAND VARIABLES. CANOR  6
      M=MP+MQ                                                          CANOR  7
      N1=1                                                             CANOR  8
      DO 105 I=1,M                                                     CANOR  9
      DO 105 J=1,M                                                     CANOR 10
      IF(I-J) 102, 103, 103                                            CANOR 11
  102 L=I+(J*J-J)/2                                                    CANOR 12
      GO TO 104                                                        CANOR 13
  103 L=J+(I*I-I)/2                                                    CANOR 14
  104 N1=N1+1                                                          CANOR 15
  105 R(N1)=RR(L)                                                      CANOR 16
      L=MP                                                             CANOR 17
      DO 108 J=2,MP                                                    CANOR 18
      N1=M*(J-1)                                                       CANOR 19
      DO 108 I=1,MP                                                    CANOR 20
      L=L+1                                                            CANOR 21
      N1=N1+1                                                          CANOR 22
  108 R(L)=R(N1)                                                       CANOR 23
      N2=MP+1                                                          CANOR 24
      L=0                                                              CANOR 25
      DO 110 J=N2,M                                                    CANOR 26
      N1=M*(J-1)                                                       CANOR 27
      DO 110 I=1,MP                                                    CANOR 28
      L=L+1                                                            CANOR 29
      N1=N1+1                                                          CANOR 30
  110 COEFL(L)=R(N1)                                                   CANOR 31
      L=0                                                              CANOR 32
      DO 120 J=N2,M                                                    CANOR 33
      N1=M*(J-1)+MP                                                    CANOR 34
      DO 120 I=N2,M                                                    CANOR 35
      L=L+1                                                            CANOR 36
      N1=N1+1                                                          CANOR 37
  120 COEFR(L)=R(N1)                                                   CANOR 38
C        SOLVE THE CANONICAL EQUATION                                  CANOR 39
      L=MP*MP+1                                                        CANOR 40
      K=L+MP                                                           CANOR 41
      CALL MINV (R,MP,DET,R(L),R(K))                                   CANOR 42
C        CALCULATE T = INVERSE OF R11 * R12                            CANOR 43
      DO 140 I=1,MP                                                    CANOR 44
      N2=0                                                             CANOR 45
      DO 130 J=1,MQ                                                    CANOR 46
      N1=I-MP                                                          CANOR 47
      ROOTS(J)=0.0                                                     CANOR 48
      DO 130 K=1,MP                                                    CANOR 49
      N1=N1+MP                                                         CANOR 50
      N2=N2+1                                                          CANOR 51
  130 ROOTS(J)=ROOTS(J)+R(N1)*COEFL(N2)                                CANOR 52
      L=I-MP                                                           CANOR 53
      DO 140 J=1,MQ                                                    CANOR 54
```

```
      L=L+MP                                                            CANOR 55
  140 R(L)=ROOTS(J)                                                     CANOR 56
C        CALCULATE A = R21 * T                                          CANOR 57
      L=MP*MQ                                                           CANOR 58
      N3=L+1                                                            CANOR 59
      DO 160 J=1,MQ                                                     CANOR 60
      N1=0                                                              CANOR 61
      DO 160 I=1,MQ                                                     CANOR 62
      N2=MP*(J-1)                                                       CANOR 63
      SUM=0.0                                                           CANOR 64
      DO 150 K=1,MP                                                     CANOR 65
      N1=N1+1                                                           CANOR 66
      N2=N2+1                                                           CANOR 67
  150 SUM=SUM+COEFL(N1)*R(N2)                                           CANOR 68
      L=L+1                                                             CANOR 69
  160 R(L)=SUM                                                          CANOR 70
C        CALCULATE EIGENVALUES WITH ASSOCIATED EIGENVECTORS OF THE      CANOR 71
C        INVERSE OF R22 * A                                             CANOR 72
      L=L+1                                                             CANOR 73
      CALL NROOT (MQ,R(N3),COEFR,ROOTS,R(L))                            CANOR 74
C        FOR EACH VALUE OF I = 1, 2, ..., MQ, CALCULATE THE FOLLOWING   CANOR 75
C        STATISTICS                                                     CANOR 76
      DO 210 I=1,MQ                                                     CANOR 77
C        TEST WHETHER EIGENVALUE IS GREATER THAN ZERO                   CANOR 78
      IF(ROOTS(I)) 220, 220, 165                                        CANOR 79
C        CANONICAL CORRELATION                                          CANOR 80
  165 CANR(I)= SQRT(ROOTS(I))                                           CANOR 81
C        CHI-SQUARE                                                     CANOR 82
      WLAM(I)=1.0                                                       CANOR 83
      DO 170 J=I,MQ                                                     CANOR 84
  170 WLAM(I)=WLAM(I)*(1.0-ROOTS(J))                                    CANOR 85
      FN=N                                                              CANOR 86
      FMP=MP                                                            CANOR 87
      FMQ=MQ                                                            CANOR 88
      BAT = WLAM(I)                                                     CANOR 89
  175 CHISQ(I) = -(FN-.5*(FMP+FMQ+1.0))*ALOG(BAT)                       CANOR 90
C        DEGREES OF FREEDOM FOR CHI-SQUARE                              CANOR 91
      N1=I-1                                                            CANOR 92
      NDF(I)=(MP-N1)*(MQ-N1)                                            CANOR 93
C        I-TH SET OF RIGHT HAND COEFFICIENTS                            CANOR 94
      N1=MQ*(I-1)                                                       CANOR 95
      N2=MQ*(I-1)+L-1                                                   CANOR 96
      DO 180 J=1,MQ                                                     CANOR 97
      N1=N1+1                                                           CANOR 98
      N2=N2+1                                                           CANOR 99
  180 COEFR(N1)=R(N2)                                                   CANOR100
C        I-TH SET OF LEFT HAND COEFFICIENTS                             CANOR101
      DO 200 J=1,MP                                                     CANOR102
      N1=J-MP                                                           CANOR103
      N2=MQ*(I-1)                                                       CANOR104
      K=MP*(I-1)+J                                                      CANOR105
      COEFL(K)=0.0                                                      CANOR106
      DO 190 JJ=1,MQ                                                    CANOR107
      N1=N1+MP                                                          CANOR108
      N2=N2+1                                                           CANOR109
  190 COEFL(K)=COEFL(K)+R(N1)*COEFR(N2)                                 CANOR110
  200 COEFL(K)=COEFL(K)/CANR(I)                                         CANOR111
  210 CONTINUE                                                          CANOR112
  220 RETURN                                                            CANOR113
      END                                                               CANOR114
```

NROOT

This subroutine calculates the eigenvalues, λ_i, and the matrix of eigenvectors, V, of a real square non-symmetric matrix of the special form $B^{-1}A$, where both B and A are real symmetric matrices and B is positive-definite. This subroutine is normally called by the subroutine CANOR in performing a canonical correlation analysis. The computational steps are as follows.

A symmetric matrix (storage mode 1) is formed by using the upper triangle elements of the square matrix B. Then, the eigenvalues, h_i, and the matrix of eigenvectors, H, of the symmetric matrix are calculated by the subroutine EIGEN.

The reciprocal of square root of each eigenvalue is formed as follows:

$$\mu_i = \frac{1}{\sqrt{h_i}} \tag{1}$$

where i = 1, 2, ..., m

m = order of matrix B

The matrix $B^{-1/2}$ is formed by multiplying the j^{th} column vector of H by μ_j, where j = 1, 2, ..., m.

The symmetric matrix $S = (B^{-1/2})' AB^{-1/2}$ is formed in the following two matrix multiplications:

$$Q = (B^{-1/2})' A \tag{2}$$

$$S = QB^{-1/2} \tag{3}$$

and eigenvalues, λ_i, and the matrix of eigenvectors, M, of S are calculated by the subroutine EIGEN.

The matrix $W = B^{-1/2}M$ is formed, and the vectors in W are normalized to form the matrix of eigenvectors, V, by the following equation:

$$V_{ij} = \frac{W_{ij}}{\sqrt{SUMV_j}} \tag{4}$$

where i = 1, 2, ..., m

j = 1, 2, ..., m

$$SUMV_j = \sum_{i=1}^{m} W_{ij}^2 \tag{5}$$

Subroutine NROOT

Purpose:

Compute 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.

Usage:

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

Description of parameters:

- 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 columnwise.

Remarks:

None.

Subroutines and function subprograms required:

EIGEN

Method:

Refer to W. W. Cooley and P. R. Lohnes, 'Multivariate Procedures for the Behavioral Sciences', John Wiley and Sons, 1962, Chapter 3.

```
      SUBROUTINE NROOT (M,A,B,XL,X)                                     NROOT  1
      DIMENSION A(1),B(1),XL(1),X(1)                                    NROOT  2
C        COMPUTE EIGENVALUES AND EIGENVECTORS OF B                      NROOT  3
      K=1                                                               NROOT  4
      DO 100 J=2,M                                                      NROOT  5
      L=M*(J-1)                                                         NROOT  6
      DO 100 I=1,J                                                      NROOT  7
      L=L+1                                                             NROOT  8
      K=K+1                                                             NROOT  9
  100 B(K)=B(L)                                                         NROOT 10
C        THE MATRIX B IS A REAL SYMMETRIC MATRIX.                       NROOT 11
      MV=0                                                              NROOT 12
      CALL EIGEN (B,X,M,MV)                                             NROOT 13
C        FORM RECIPROCALS OF SQUARE ROOT OF EIGENVALUES.  THE RESULTS   NROOT 14
C        ARE PREMULTIPLIED BY THE ASSOCIATED EIGENVECTORS.              NROOT 15
      L=0                                                               NROOT 16
      DO 110 J=1,M                                                      NROOT 17
      L=L+J                                                             NROOT 18
  110 XL(J)=1.0/ SQRT( ABS(B(L)))                                       NROOT 19
      K=0                                                               NROOT 20
      DO 115 J=1,M                                                      NROOT 21
      DO 115 I=1,M                                                      NROOT 22
      K=K+1                                                             NROOT 23
  115 B(K)=X(K)*XL(J)                                                   NROOT 24
C        FORM (B**(-1/2))PRIME * A * (B**(-1/2))                        NROOT 25
      DO 120 I=1,M                                                      NROOT 26
      N2=0                                                              NROOT 27
      DO 120 J=1,M                                                      NROOT 28
      N1=M*(I-1)                                                        NROOT 29
      L=M*(J-1)+I                                                       NROOT 30
      X(L)=0.0                                                          NROOT 31
      DO 120 K=1,M                                                      NROOT 32
      N1=N1+1                                                           NROOT 33
      N2=N2+1                                                           NROOT 34
  120 X(L)=X(L)+B(N1)*A(N2)                                             NROOT 35
      L=0                                                               NROOT 36
      DO 130 J=1,M                                                      NROOT 37
      DO 130 I=1,J                                                      NROOT 38
      N1=I-M                                                            NROOT 39
      N2=M*(J-1)                                                        NROOT 40
      L=L+1                                                             NROOT 41
      A(L)=0.0                                                          NROOT 42
      DO 130 K=1,M                                                      NROOT 43
      N1=N1+M                                                           NROOT 44
      N2=N2+1                                                           NROOT 45
  130 A(L)=A(L)+X(N1)*B(N2)                                             NROOT 46
C        COMPUTE EIGENVALUES AND EIGENVECTORS OF A                      NROOT 47
      CALL EIGEN (A,X,M,MV)                                             NROOT 48
      L=0                                                               NROOT 49
      DO 140 I=1,M                                                      NROOT 50
      L=L+I                                                             NROOT 51
  140 XL(I)=A(L)                                                        NROOT 52
C        COMPUTE THE NORMALIZED EIGENVECTORS                            NROOT 53
      DO 150 I=1,M                                                      NROOT 54
      N2=0                                                              NROOT 55
      DO 150 J=1,M                                                      NROOT 56
      N1=I-M                                                            NROOT 57
      L=M*(J-1)+I                                                       NROOT 58
      A(L)=0.0                                                          NROOT 59
      DO 150 K=1,M                                                      NROOT 60
      N1=N1+M                                                           NROOT 61
      N2=N2+1                                                           NROOT 62
  150 A(L)=A(L)+B(N1)*X(N2)                                             NROOT 63
      L=0                                                               NROOT 64
      K=0                                                               NROOT 65
      DO 180 J=1,M                                                      NROOT 66
      SUMV=0.0                                                          NROOT 67
      DO 170 I=1,M                                                      NROOT 68
      L=L+1                                                             NROOT 69
  170 SUMV=SUMV+A(L)*A(L)                                               NROOT 70
  175 SUMV= SQRT(SUMV)                                                  NROOT 71
      DO 180 I=1,M                                                      NROOT 72
      K=K+1                                                             NROOT 73
  180 X(K)=A(K)/SUMV                                                    NROOT 74
      RETURN                                                            NROOT 75
      END                                                               NROOT 76
```

Statistics - Analysis of Variance

In the Scientific Subroutine Package, analysis of variance is normally performed by calling the following three subroutines in sequence:

1. AVDAT - to place data in properly distributed positions of storage
2. AVCAL - to apply the operators sigma and delta in order to compute deviates for analysis of variance
3. MEANQ - to pool the deviates and compute sums of squares, degrees of freedom, and mean squares

AVDAT

This subroutine places data for analysis of variance in properly distributed positions of storage.

The size of data array X, required for an analysis of variance problem, is calculated as follows:

$$n = \prod_{i=1}^{k} (L_i + 1) \qquad (1)$$

where L_i = number of levels of i^{th} factor

k = number of factors

The input data placed in the lower part of the array X are moved temporarily to the upper part of the array. From there, the data are redistributed according to the equation (4) below. Prior to that, multipliers, s_j, to be used in finding proper positions of storage, are calculated as follows:

$$s_1 = 1 \qquad (2)$$

$$s_j = \prod_{i=1}^{j-1} (L_i + 1) \qquad (3)$$

where J = 2, 3, ..., k

Then, a position for each data point is calculated by the following equation:

$$S = KOUNT_1 + \sum_{j=2}^{k} s_j \cdot (KOUNT_j - 1) \qquad (4)$$

where $KOUNT_j$ = value of j^{th} subscript of the data to be stored.

The subroutine increments the value(s) of subscript(s) after each data point is stored.

Subroutine AVDAT

Purpose:

Place data for analysis of variance in properly distributed positions of storage. This subroutine is normally followed by calls to AVCAL and MEANQ subroutines in the performance of analysis of variance for a complete factorial design.

Usage:

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

Description of parameters:

- 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 in.
- X - When the subroutine 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 one to each level of variable and (2) obtaining the cumulative product of all levels. (The length of X = (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-variable analysis of variance design, where one variable has 3 levels and the other two variables have 2 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 inner subscript, namely I, changes first. When I=3, the next inner subscript, J, changes and so on until I=3, J=2, and K=2.

Subroutines and function subprograms required:

None.

Method:

The method is based on the technique discussed by H. D. Hartley in 'Mathematical Methods for Digital Computers', edited by A. Ralston and H. Wilf, John Wiley and Sons, 1962, Chapter 20.

```
      SUBROUTINE AVDAT (K,LEVEL,N,X,L,ISTEP,KOUNT)                          AVDAT  1
      DIMENSION LEVEL(1),X(1),ISTEP(1),KOUNT(1)                             AVDAT  2
C        CALCULATE TOTAL DATA AREA REQUIRED                                 AVDAT  3
      M=LEVEL(1)+1                                                          AVDAT  4
      DO 105 I=2,K                                                          AVDAT  5
  105 M=M*(LEVEL(I)+1)                                                      AVDAT  6
C        MOVE DATA TO THE UPPER PART OF THE ARRAY X                         AVDAT  7
C        FOR THE PURPOSE OF REARRANGEMENT                                   AVDAT  8
      N1=M+1                                                                AVDAT  9
      N2=N+1                                                                AVDAT 10
      DO 107 I=1,N                                                          AVDAT 11
      N1=N1-1                                                               AVDAT 12
      N2=N2-1                                                               AVDAT 13
  107 X(N1)=X(N2)                                                           AVDAT 14
C        CALCULATE MULTIPLIERS TO BE USED IN FINDING STORAGE LOCATIONS      AVDAT 15
C        FOR INPUT DATA                                                     AVDAT 16
      ISTEP(1)=1                                                            AVDAT 17
      DO 110 I=2,K                                                          AVDAT 18
  110 ISTEP(I)=ISTEP(I-1)*(LEVEL(I-1)+1)                                    AVDAT 19
      DO 115 I=1,K                                                          AVDAT 20
  115 KOUNT(I)=1                                                            AVDAT 21
C        PLACE DATA IN PROPER LOCATIONS                                     AVDAT 22
      N1=N1-1                                                               AVDAT 23
      DO 135 I=1,N                                                          AVDAT 24
      L=KOUNT(1)                                                            AVDAT 25
      DO 120 J=2,K                                                          AVDAT 26
  120 L=L+ISTEP(J)*(KOUNT(J)-1)                                             AVDAT 27
      N1=N1+1                                                               AVDAT 28
      X(L)=X(N1)                                                            AVDAT 29
      DO 130 J=1,K                                                          AVDAT 30
      IF(KOUNT(J)-LEVEL(J)) 124, 125, 124                                   AVDAT 31
  124 KOUNT(J)=KOUNT(J)+1                                                   AVDAT 32
      GO TO 135                                                             AVDAT 33
  125 KOUNT(J)=1                                                            AVDAT 34
  130 CONTINUE                                                              AVDAT 35
  135 CONTINUE                                                              AVDAT 36
      RETURN                                                                AVDAT 37
      END                                                                   AVDAT 38
```

AVCAL

This subroutine performs the calculus for the general k-factor experiment: operator Σ and operator Δ. An example is presented in terms of k=3 to illustrate these operators.

Let x_{abc} denote the experimental reading from the a^{th} level of factor A, the b^{th} level of factor B, and the c^{th} level of factor C. The symbols A, B, and C will also denote the number of levels for each factor so that a = 1, 2, ..., A; b = 1, 2, ..., B; and c = 1, 2, ..., C.

With regard to the first factor A,

operator $\sum_a \equiv$ sum over all levels a = 1, 2, ..., A, holding the other subscripts at constant levels, and

operator $\underset{a}{\Delta} \equiv$ multiply all items by A and subtract the result of $\sum_a$ from all items

In mathematical notations, these operators are defined as follows:

$$\sum_a x_{abc} \equiv X_{.bc} \equiv \sum_{a=1}^{A} x_{abc} \qquad (1)$$

$$\underset{a}{\Delta}\, x_{abc} \equiv A\, x_{abc} - X_{.bc} \qquad (2)$$

The operators Σ and Δ will be applied sequentially with regard to all factors A, B, and C. Upon the completion of these operators, the storage array X contains deviates to be used for analysis of variance components in the subroutine MEANQ.

Subroutine AVCAL

Purpose:

Perform the calculus of a factorial experiment using operator sigma and operator delta. This subroutine is preceded by subroutine ADVAT and followed by subroutine MEANQ in the performance of analysis of variance for a complete factorial design.

Usage:

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

Description of parameters:

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 have 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.

Subroutines and function subprograms required:

None.

Method:

The method is based on the technique discussed by H. O. Hartley in 'Mathematical Methods for Digital Computers', edited by A. Ralston and H. Wilf, John Wiley and Sons, 1962, Chapter 20.

```
      SUBROUTINE AVCAL (K,LEVEL,X,L,ISTEP,LASTS)                        AVCAL  1
      DIMENSION LEVEL(1),X(1),ISTEP(1),LASTS(1)                         AVCAL  2
C        CALCULATE THE LAST DATA POSITION OF EACH FACTOR                AVCAL  3
      LASTS(1)=L+1                                                      AVCAL  4
      DO 145 I=2,K                                                      AVCAL  5
  145 LASTS(I)=LASTS(I-1)+ISTEP(I)                                      AVCAL  6
C        PERFORM CALCULUS OF OPERATION                                  AVCAL  7
  150 DO 175 I=1,K                                                      AVCAL  8
      L=1                                                               AVCAL  9
      LL=1                                                              AVCAL 10
      SUM=0.0                                                           AVCAL 11
      NN=LEVEL(I)                                                       AVCAL 12
      FN=NN                                                             AVCAL 13
      INCRE=ISTEP(I)                                                    AVCAL 14
      LAST=LASTS(I)                                                     AVCAL 15
C        SIGMA OPERATION                                                AVCAL 16
  155 DO 160 J=1,NN                                                     AVCAL 17
      SUM=SUM+X(L)                                                      AVCAL 18
  160 L=L+INCRE                                                         AVCAL 19
      X(L)=SUM                                                          AVCAL 20
C        DELTA OPERATION                                                AVCAL 21
      DO 165 J=1,NN                                                     AVCAL 22
      X(LL)=FN*X(LL)-SUM                                                AVCAL 23
  165 LL=LL+INCRE                                                       AVCAL 24
      SUM=0.0                                                           AVCAL 25
      IF(L-LAST) 167, 175, 175                                          AVCAL 26
  167 IF(L-LAST+INCRE) 168, 168, 170                                    AVCAL 27
  168 L=L+INCRE                                                         AVCAL 28
      LL=LL+INCRE                                                       AVCAL 29
      GO TO 155                                                         AVCAL 30
  170 L=L+INCRE+1-LAST                                                  AVCAL 31
      LL=LL+INCRE+1-LAST                                                AVCAL 32
      GO TO 155                                                         AVCAL 33
  175 CONTINUE                                                          AVCAL 34
      RETURN                                                            AVCAL 35
      END                                                               AVCAL 36
```

MEANQ

This subroutine performs the mean square operation for the general k-factor experiment in the following two steps:

1. Square each value of deviates for analysis of variance stored in the array X (the result of the operators Σ and Δ applied in the subroutine AVCAI).

2. Add the squared value into summation storage. In a three-factor experiment, for example, the squared value is added into one of seven storages ($7 = 2^3 - 1$) as shown in the first column of Table 1. The symbols A, B, and C in the first column denote factor A, factor B, and factor C.

After the mean square operation is completed for all values in the storage array X, the subroutine forms sums of squares of analysis of variance by dividing the totals of squared values by proper divisors. These divisors for the three-factor experiment mentioned above are shown in the second column of Table 1. The symbols A, B, and C in the second column denote the number of levels for each factor.

The subroutine, then, forms mean squares by dividing sums of squares by degrees of freedom. The third column of the summary table shows the degrees of freedom. The symbols A, B, and C denote the number of levels for each factor.

Table 1. Table Showing Summation Storages, Divisors to Form Sum of Squares, and Degrees of Freedom (Subroutine MEANQ)

Designation of Store and of Quantity Contained in it	Divisor Required to Form Sum of Squares of Analysis of Variance	Degrees of Freedom Required to Form Mean Squares
$(A)^2$	ABC·A	(A-1)
$(B)^2$	ABC·B	(B-1)
$(AB)^2$	ABC·AB	(A-1) (B-1)
$(C)^2$	ABC·C	(C-1)
$(AC)^2$	ABC·AC	(A-1) (C-1)
$(BC)^2$	ABC·BC	(B-1) (C-1)
$(ABC)^2$	ABC·ABC	(A-1) (B-1) (C-1)

Subroutine MEANQ

Purpose:

Compute sum of squares, degrees of freedom, and mean square using the mean square operator. This subroutine normally follows calls to AVDAT and AVCAL subroutines in the performance of analysis of variance for a complete factorial design.

Usage:

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

Description of parameters:

- 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 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 K^{th} power minus one, (2**K)-1.
- NDF - Output vector containing degrees of freedom. The length of NDF is 2 to the K^{th} power minus one, (2**K)-1.
- SMEAN - Output vector containing mean squares. The length of SMEAN is 2 to the K^{th} power minus one, (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.

Subroutines and function subprograms required:

None.

Method:

The method is based on the technique discussed by H. O. Hartley in 'Mathematical Methods for Digital Computers', edited by A. Ralston and H. Wilf, John Wiley and Sons, 1962, Chapter 20.

```
      SUBROUTINE MEANQ (K,LEVEL,X,GMEAN,SUMSQ,NDF,SMEAN,MSTEP,KOUNT,     MEANQ  1
     1                  LASTS)                                           MEANQ  2
      DIMENSION LEVEL(1),X(1),SUMSQ(1),NDF(1),SMEAN(1),MSTEP(1),         MEANQ  3
     1          KOUNT(1),LASTS(1)                                        MEANQ  4
C        CALCULATE TOTAL NUMBER OF DATA                                  MEANQ  5
      N=LEVEL(1)                                                         MEANQ  6
      DO 150 I=2,K                                                       MEANQ  7
  150 N=N*LEVEL(I)                                                       MEANQ  8
C        SET UP CONTROL FOR MEAN SQUARE OPERATOR                         MEANQ  9
      LASTS(1)=LEVEL(1)                                                  MEANQ 10
      DO 178 I=2,K                                                       MEANQ 11
  178 LASTS(I)=LEVEL(I)+1                                                MEANQ 12
      NN=1                                                               MEANQ 13
C        CLEAR THE AREA TO STORE SUMS OF SQUARES                         MEANQ 14
      LL=(2**K)-1                                                        MEANQ 15
      MSTEP(1)=1                                                         MEANQ 16
      DO 180 I=2,K                                                       MEANQ 17
  180 MSTEP(I)=MSTEP(I-1)*2                                              MEANQ 18
      DO 185 I=1,LL                                                      MEANQ 19
  185 SUMSQ(I)=0.0                                                       MEANQ 20
C        PERFORM MEAN SQUARE OPERATOR                                    MEANQ 21
      DO 190 I=1,K                                                       MEANQ 22
  190 KOUNT(I)=0                                                         MEANQ 23
  200 L=0                                                                MEANQ 24
      DO 260 I=1,K                                                       MEANQ 25
      IF(KOUNT(I)-LASTS(I)) 210, 250, 210                                MEANQ 26
  210 IF(L) 220, 220, 240                                                MEANQ 27
  220 KOUNT(I)=KOUNT(I)+1                                                MEANQ 28
      IF(KOUNT(I)-LEVEL(I)) 230, 230, 250                                MEANQ 29
  230 L=L+MSTEP(I)                                                       MEANQ 30
      GO TO 260                                                          MEANQ 31
  240 IF(KOUNT(I)-LEVEL(I)) 230, 260, 230                                MEANQ 32
  250 KOUNT(I)=0                                                         MEANQ 33
  260 CONTINUE                                                           MEANQ 34
      IF(L) 285, 285, 270                                                MEANQ 35
  270 SUMSQ(L)=SUMSQ(L)+X(NN)*X(NN)                                      MEANQ 36
      NN=NN+1                                                            MEANQ 37
      GO TO 200                                                          MEANQ 38
C        CALCULATE THE GRAND MEAN                                        MEANQ 39
  285 FN=N                                                               MEANQ 40
      GMEAN=X(NN)/FN                                                     MEANQ 41
C        CALCULATE FIRST DIVISOR REQUIRED TO FORM SUM OF SQUARES AND     MEANQ 42
C        DIVISOR, WHICH IS EQUAL TO DEGREES OF FREEDOM, REQUIRED TO FORMMEANQ 43
C        MEAN SQUARES                                                    MEANQ 44
      DO 310 I=2,K                                                       MEANQ 45
  310 MSTEP(I)=0                                                         MEANQ 46
      NN=0                                                               MEANQ 47
      MSTEP(1)=1                                                         MEANQ 48
  320 ND1=1                                                              MEANQ 49
      ND2=1                                                              MEANQ 50
      DO 340 I=1,K                                                       MEANQ 51
      IF(MSTEP(I)) 330, 340, 330                                         MEANQ 52
  330 ND1=ND1*LEVEL(I)                                                   MEANQ 53
      ND2=ND2*(LEVEL(I)-1)                                               MEANQ 54
  340 CONTINUE                                                           MEANQ 55
      FN1=ND1                                                            MEANQM01
      FN1=FN*FN1                                                         MEANQM02
      FN2=ND2                                                            MEANQ 57
      NN=NN+1                                                            MEANQ 58
      SUMSQ(NN)=SUMSQ(NN)/FN1                                            MEANQ 59
      NDF(NN)=ND2                                                        MEANQ 60
      SMEAN(NN)=SUMSQ(NN)/FN2                                            MEANQ 61
      IF(NN-LL) 345, 370, 370                                            MEANQ 62
  345 DO 360 I=1,K                                                       MEANQ 63
      IF(MSTEP(I)) 347, 350, 347                                         MEANQ 64
  347 MSTEP(I)=0                                                         MEANQ 65
      GO TO 360                                                          MEANQ 66
  350 MSTEP(I)=1                                                         MEANQ 67
      GO TO 320                                                          MEANQ 68
  360 CONTINUE                                                           MEANQ 69
  370 RETURN                                                             MEANQ 70
      END                                                                MEANQ 71
```

Statistics - Discriminant Analysis

In the Scientific Subroutine Package, discriminant analysis is normally performed by calling the following three subroutines in sequence:

1. DMATX - to compute means of variables in each group and a pooled dispersion matrix
2. MINV - to invert the pooled dispersion matrix
3. DISCR - to compute coefficients of discriminant functions and evaluate the functions for each observation (individual)

DMATX

This subroutine calculates means of variables in each group and a pooled dispersion matrix for the set of groups in a discriminant analysis.

For each group k = 1, 2, ..., g, the subroutine calculates means and sums of cross-products of deviations from means as follows:

Means:

$$\bar{x}_{jk} = \frac{\sum_{i=1}^{n_k} x_{ijk}}{n_k} \qquad (1)$$

where n_k = sample size in the k^{th} group

j = 1, 2, ..., m are variables

Sum of cross-products of deviations from means:

$$S_k = \left\{ s^k_{j\ell} \right\} = \sum (x_{ijk} - \bar{x}_{jk})\,(x_{i\ell k} - \bar{x}_{\ell k}) \qquad (2)$$

where j = 1, 2, ..., m

ℓ = 1, 2, ..., m

The pooled dispersion matrix is calculated as follows:

$$D = \frac{\sum_{k=1}^{g} S_k}{\sum_{k=1}^{g} n_k - g} \qquad (3)$$

where g = number of groups

Subroutine DMATX

Purpose:

Compute means of variables in each group and a pooled dispersion matrix for all the groups. Normally this subroutine is used in the performance of discriminant analysis.

Usage:

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

Description of parameters:

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), etc. The first subscript is case number, the second subscript is variable number and the third subscript is 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.

Subroutines and function subprograms required:

None.

Method:

Refer to 'BMD Computer Programs Manual', edited by W. J. Dixon, UCLA, 1964, and T. W. Anderson, 'Introduction to Multivariate Statistical Analysis', John Wiley and Sons, 1958, Section 6.6-6.8.

```
      SUBROUTINE DMATX (K,M,N,X,XBAR,D,CMEAN)                          DMATX  1
      DIMENSION N(1),X(1),XBAR(1),D(1),CMEAN(1)                        DMATX  2
      MM=M*M                                                           DMATX  3
      DO 100 I=1,MM                                                    DMATX  4
  100 D(I)=0.0                                                         DMATX  5
C         CALCULATE MEANS                                            DMATX  6
      N4=0                                                             DMATX  7
      L=0                                                              DMATX  8
      LM=0                                                             DMATX  9
      DO 160 NG=1,K                                                    DMATX 10
      N1=N(NG)                                                         DMATX 11
      FN=N1                                                            DMATX 12
      DO 130 J=1,M                                                     DMATX 13
      LM=LM+1                                                          DMATX 14
      XBAR(LM)=0.0                                                     DMATX 15
      DO 120 I=1,N1                                                    DMATX 16
      L=L+1                                                            DMATX 17
  120 XBAR(LM)=XBAR(LM)+X(L)                                           DMATX 18
  130 XBAR(LM)=XBAR(LM)/FN                                             DMATX 19
C         CALCULATE SUMS OF CROSS-PRODUCTS OF DEVIATIONS               DMATX 20
      LMEAN=LM-M                                                       DMATX 21
      DO 150 I=1,N1                                                    DMATX 22
      LL=N4+I-N1                                                       DMATX 23
      DO 140 J=1,M                                                     DMATX 24
      LL=LL+N1                                                         DMATX 25
      N2=LMEAN+J                                                       DMATX 26
  140 CMEAN(J)=X(LL)-XBAR(N2)                                          DMATX 27
      LL=0                                                             DMATX 28
      DO 150 J=1,M                                                     DMATX 29
      DO 150 JJ=1,M                                                    DMATX 30
      LL=LL+1                                                          DMATX 31
  150 D(LL)=D(LL)+CMEAN(J)*CMEAN(JJ)                                   DMATX 32
  160 N4=N4+N1*M                                                       DMATX 33
C         CALCULATE THE POOLED DISPERSION MATRIX                       DMATX 34
      LL=-K                                                            DMATX 35
      DO 170 I=1,K                                                     DMATX 36
  170 LL=LL+N(I)                                                       DMATX 37
      FN=LL                                                            DMATX 38
      DO 180 I=1,MM                                                    DMATX 39
  180 D(I)=D(I)/FN                                                     DMATX 40
      RETURN                                                           DMATX 41
      END                                                              DMATX 42
```

DISCR

This subroutine performs a discriminant analysis by calculating a set of linear functions that serve as indices for classifying an individual into one of K groups.

For all groups combined, the following are obtained:

Common means:

$$\bar{X}_j = \frac{\sum_{k=1}^{g} n_k \bar{x}_{jk}}{\sum_{k=1}^{g} n_k} \quad (1)$$

where g = number of groups

j = 1, 2, ..., m are variables

n_k = sample size in the k^{th} group

$\bar{x}_{jk}$ = mean of j^{th} variable in k^{th} group

Generalized Mahalanobis D^2 statistics, V:

$$V = \sum_{i=1}^{m}\sum_{j=1}^{m} d_{ij} \sum_{k=1}^{g} n_k (\bar{x}_{ik} - \bar{X}_i)(\bar{x}_{jk} - \bar{X}_j) \quad (2)$$

where d_{ij} = the inverse element of the pooled dispersion matrix D

V can be used as chi-square (under assumption of normality) with m(g-1) degrees of freedom to test the hypothesis that the mean values are the same in all the g groups for these m variables.

For each discriminant function k* = 1, 2, ..., g, the following statistics are calculated:

Coefficients:

$$C_{ik*} = \sum_{j=1}^{m} d_{ij} \bar{x}_{jk} \quad (3)$$

where i = 1, 2, ..., m

k = k*

Constant:

$$C_{ok*} = -1/2 \sum_{j=1}^{m} \sum_{l=1}^{m} d_{jl} \bar{x}_{jk} \bar{x}_{lk} \quad (4)$$

For each i^{th} case in each k^{th} group, the following calculations are performed:

Discriminant functions:

$$f_{k*} = \sum_{j=1}^{m} C_{jk} x_{ijk} + C_{ok*} \quad (5)$$

where k* = 1, 2, ..., g

Probability associated with largest discriminant function:

$$P_L = \frac{1}{\sum_{k*=1}^{g} e^{(f_{k*} - f_L)}} \quad (6)$$

where f_L = the value of the largest discriminant function

L = the subscript of the largest discriminant function

Subroutine DISCR

Purpose:

Compute a set of linear functions which serve as indices for classifying an individual into one of several groups. Normally this subroutine is used in the performance of discriminant analysis.

Usage:

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

Description of parameters:

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.

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), etc. The first

subscript is case number, the second subscript is variable number and the third subscript is 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 functions of all cases in all groups. Calculated results are stored in the manner equivalent to a 2-dimensional area (the first subscript is case number, and the second subscript is group number). Vector P has 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.

Subroutines and function subprograms required:

None.

Method:

Refer to 'BMD Computer Programs Manual', edited by W. J. Dixon, UCLA, 1964, and T. W. Anderson, 'Introduction to Multivariate Statistical Analysis', John Wiley and Sons, 1958.

```
      SUBROUTINE DISCR (K,M,N,X,XBAR,D,CMEAN,V,C,P,LG)                  DISCR  1
      DIMENSION N(1),X(1),XBAR(1),D(1),CMEAN(1),C(1),P(1),LG(1)         DISCR  2
C        CALCULATE COMMON MEANS                                         DISCR  3
      N1=N(1)                                                           DISCR  4
      DO 100 I=2,K                                                      DISCR  5
  100 N1=N1+N(I)                                                        DISCR  6
      FNT=N1                                                            DISCR  7
      DO 110 I=1,K                                                      DISCR  8
  110 P(I)=N(I)                                                         DISCR  9
      DO 130 I=1,M                                                      DISCR 10
      CMEAN(I)=0                                                        DISCR 11
      N1=I-M                                                            DISCR 12
      DO 120 J=1,K                                                      DISCR 13
      N1=N1+M                                                           DISCR 14
  120 CMEAN(I)=CMEAN(I)+P(J)*XBAR(N1)                                   DISCR 15
  130 CMEAN(I)=CMEAN(I)/FNT                                             DISCR 16
C        CALCULATE GENERALIZED MAHALANOBIS D SQUARE                     DISCR 17
      L=0                                                               DISCR 18
      DO 140 I=1,K                                                      DISCR 19
      DO 140 J=1,M                                                      DISCR 20
      L=L+1                                                             DISCR 21
  140 C(L)=XBAR(L)-CMEAN(J)                                             DISCR 22
      V=0.0                                                             DISCR 23
      L=0                                                               DISCR 24
      DO 160 J=1,M                                                      DISCR 25
      DO 160 I=1,M                                                      DISCR 26
      N1=I-M                                                            DISCR 27
      N2=J-M                                                            DISCR 28
      SUM=0.0                                                           DISCR 29
      DO 150 IJ=1,K                                                     DISCR 30
      N1=N1+M                                                           DISCR 31
      N2=N2+M                                                           DISCR 32
  150 SUM=SUM+P(IJ)*C(N1)*C(N2)                                         DISCR 33
      L=L+1                                                             DISCR 34
  160 V=V+D(L)*SUM                                                      DISCR 35
C        CALCULATE THE COEFFICIENTS OF DISCRIMINANT FUNCTIONS           DISCR 36
      N2=0                                                              DISCR 37
      DO 190 KA=1,K                                                     DISCR 38
      DO 170 I=1,M                                                      DISCR 39
      N2=N2+1                                                           DISCR 40
  170 P(I)=XBAR(N2)                                                     DISCR 41
      IQ=(M+1)*(KA-1)+1                                                 DISCR 42
      SUM=0.0                                                           DISCR 43
      DO 180 J=1,M                                                      DISCR 44
      N1=J-M                                                            DISCR 45
      DO 180 L=1,M                                                      DISCR 46
      N1=N1+M                                                           DISCR 47
  180 SUM=SUM+D(N1)*P(J)*P(L)                                           DISCR 48
      C(IQ)=-(SUM/2.0)                                                  DISCR 49
      DO 190 I=1,M                                                      DISCR 50
      N1=I-M                                                            DISCR 51
      IQ=IQ+1                                                           DISCR 52
      C(IQ)=0.0                                                         DISCR 53
      DO 190 J=1,M                                                      DISCR 54
      N1=N1+M                                                           DISCR 55
  190 C(IQ)=C(IQ)+D(N1)*P(J)                                            DISCR 56
C        FOR EACH CASE IN EACH GROUP, CALCULATE..                       DISCR 57
C        DISCRIMINANT FUNCTIONS                                         DISCR 58
      LBASE=0                                                           DISCR 59
      N1=0                                                              DISCR 60
      DO 270 KG=1,K                                                     DISCR 61
      NN=N(KG)                                                          DISCR 62
      DO 260 I=1,NN                                                     DISCR 63
      L=I-NN+LBASE                                                      DISCR 64
      DO 200 J=1,M                                                      DISCR 65
      L=L+NN                                                            DISCR 66
  200 D(J)=X(L)                                                         DISCR 67
      N2=0                                                              DISCR 68
      DO 220 KA=1,K                                                     DISCR 69
      N2=N2+1                                                           DISCR 70
      SUM=C(N2)                                                         DISCR 71
      DO 210 J=1,M                                                      DISCR 72
      N2=N2+1                                                           DISCR 73
  210 SUM=SUM+C(N2)*D(J)                                                DISCR 74
  220 XBAR(KA)=SUM                                                      DISCR 75
C        THE LARGEST DISCRIMINANT FUNCTION                              DISCR 76
      L=1                                                               DISCP 77
      SUM=XBAR(1)                                                       DISCR 78
      DO 240 J=2,K                                                      DISCP 79
      IF(SUM-XBAR(J)) 230, 240, 240                                     DISCP 80
  230 L=J                                                               DISCP 81
      SUM=XBAR(J)                                                       DISCP 82
  240 CONTINUE                                                          DISCR 83
C        PROBABILITY ASSOCIATED WITH THE LARGEST DISCRIMINANT FUNCTION  DISCR 84
      PL=0.0                                                            DISCP 85
      DO 250 J=1,K                                                      DISCR 86
  250 PL=PL+ EXP(XBAR(J)-SUM)                                           DISCR 87
      N1=N1+L                                                           DISCR 88
      LG(N1)=L                                                          DISCR 89
  260 P(N1)=1.0/PL                                                      DISCP 90
  270 LBASE=LBASE+NN*M                                                  DISCR 91
      RETURN                                                            DISCR 92
      END                                                               DISCR 93
```

Statistics - Factor Analysis

Factor analysis is a method of analyzing the intercorrelations within a set of variables. It determines whether the variance in the original set of variables can be accounted for adequately by a smaller number of basic categories, namely factors.

In the Scientific Subroutine Package, factor analysis is normally performed by calling the following five subroutines in sequence:

1. CORRE - to find means, standard deviations, and correlation matrix
2. EIGEN - to compute eigenvalues and associated eigenvectors of the correlation matrix
3. TRACE - to select the eigenvalues that are greater than or equal to the control value specified by the user
4. LOAD - to compute a factor matrix
5. VARMX - to perform varimax rotation of the factor matrix

The subroutine CORRE works in either of two ways: (1) it expects all observations in core, or (2) it triggers a user-provided input subroutine, DATA, to read one observation at a time into a work area. In either case, the user must provide a subroutine named DATA (see "Subroutines Required" in the description of subroutine CORRE).

TRACE

This subroutine finds k, the number of eigenvalues that are greater than or equal to the value of a specified constant. The given eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$ must be arranged in descending order.

Cumulative percentage for these k eigenvalues are:

$$d_j = \sum_{i=1}^{j} \frac{\lambda_i}{m} \qquad (1)$$

where $j = 1, 2, \ldots, k$

m = number of eigenvalues (or variables)

$k \leq m$

Subroutine TRACE

Purpose:

Compute cumulative percentage of eigenvalues greater than or equal to a constant specified by the user. This subroutine normally occurs in a sequence of calls to subroutines CORRE, EIGEN, TRACE, LOAD, and VARMX in the performance of a factor analysis.

Usage:

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

Description of parameters:

- M - Number of variables.
- R - Input matrix (symmetric and stored in compressed form with only upper triangle by column in core) 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. Cumulative percentage of eigenvalues which are greater than or equal to this value is calculated.
- 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 which are greater than or equal to CON.

Remarks:
None.

Subroutines and function subprograms required:
None.

Method:
Each eigenvalue greater than or equal to CON is divided by M and the result is added to the previous total to obtain the cumulative percentage for each eigenvalue.

```
      SUBROUTINE TRACE (M,R,CON,K,D)                                    TRACE  1
      DIMENSION R(1),D(1)                                               TRACE  2
      FM=M                                                              TRACE  3
      L=0                                                               TRACE  4
      DO 100 I=1,M                                                      TRACE  5
      L=L+I                                                             TRACE  6
  100 D(I)=R(L)                                                         TRACE  7
      K=0                                                               TRACE  8
C        TEST WHETHER I-TH EIGENVALUE IS GREATER                        TRACE  9
C        THAN OR EQUAL TO THE CONSTANT                                  TRACE 10
      DO 110 I=1,M                                                      TRACE 11
      IF(D(I)-CON) 120, 105, 105                                        TRACE 12
  105 K=K+1                                                             TRACE 13
  110 D(I)=D(I)/FM                                                      TRACE 14
C        COMPUTE CUMULATIVE PERCENTAGE OF EIGENVALUES                   TRACE 15
  120 DO 130 I=2,K                                                      TRACE 16
  130 D(I)=D(I)+D(I-1)                                                  TRACE 17
      RETURN                                                            TRACE 18
      END                                                               TRACE 19
```

LOAD

This subroutine calculates the coefficients of each factor by multiplying the elements of each normalized eigenvector by the square root of the corresponding eigenvalue.

$$a_{ij} = v_{ij} \cdot \sqrt{\lambda_j} \qquad (1)$$

where i = 1, 2, ..., m are variables

j = 1, 2, ..., k are eigenvalues retained (see the subroutine TRACE)

$k \leq m$

Subroutine LOAD

Purpose:
Compute 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.

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

Description of parameters:
M - Number of variables.
K - Number of factors.
R - A matrix (symmetric and stored in compressed form with only upper triangle by column in core) 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.)
V - When this subroutine is called, matrix V (M by M) contains eigenvectors columnwise. Upon returning to the calling program, matrix V contains a factor matrix (M by K).

Remarks:
None.

Subroutines and function subprograms required:
None.

Method:

Normalized eigenvectors are converted to the factor pattern by multiplying the elements of each vector by the square root of the corresponding eigenvalue.

```
      SUBROUTINE LOAD (M,K,R,V)                                  LOAD   1
      DIMENSION R(1),V(1)                                        LOAD   2
      L=0                                                        LOAD   3
      JJ=0                                                       LOAD   4
      DO 160 J=1,K                                               LOAD   5
      JJ=JJ+J                                                    LOAD   6
  150 SQ= SQRT(R(JJ))                                            LOAD   7
      DO 160 I=1,M                                               LOAD   8
      L=L+1                                                      LOAD   9
  160 V(L)=SQ*V(L)                                               LOAD  10
      RETURN                                                     LOAD  11
      END                                                        LOAD  12
```

VARMX

This subroutine performs orthogonal rotations on a m by k factor matrix such that:

$$\sum_j \left\{ m \sum_i \left(a_{ij}^2 / h_i^2 \right)^2 - \left[\sum_i \left(a_{ij}^2 / h_i^2 \right) \right] \right\}^2 \qquad (1)$$

is a maximum, where i = 1, 2, ..., m are variables, j = 1, 2, ..., k are factors, a_{ij} is the loading for the i^{th} variable on the j^{th} factor, and h_i^2 is the communality of the i^{th} variable defined below.

Communalities:

$$h_i^2 = \sum_{j=1}^{k} a_{ij}^2 \qquad (2)$$

where i = 1, 2, ..., m

Normalized factor matrix:

$$b_{ij} = a_{ij} / \sqrt{h_i^2} \qquad (3)$$

where i = 1, 2, ..., m

j = 1, 2, ..., k

Variance for factor matrix:

$$V_c = \sum_j \left\{ \left[m \sum_i \left(b_{ij}^2 \right)^2 - \left(\sum_i b_{ij}^2 \right)^2 \right] / m^2 \right\} \qquad (4)$$

where c = 1, 2, ... (iteration cycle)

Convergence test:

$$\text{If } V_c - V_{c-1} \leq 10^{-7} \qquad (5)$$

four successive times, the program stops rotation and performs the equation (28). Otherwise, the program repeats rotation of factors until the convergence test is satisfied.

Rotation of two factors:

The subroutine rotates two normalized factors (b_{ij}) at a time. 1 with 2, 1 with 3, ..., 1 with k, 2 with 3, ..., 2 with k, ..., k - 1 with k. This constitutes one iteration cycle.

Assume that x and y are factors to be rotated, where x is the lower-numbered or left-hand factor, the following notation for rotating these two factors is used:

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ x_m & y_m \end{bmatrix} \cdot \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} = \begin{bmatrix} X_1 & Y_1 \\ X_2 & Y_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ X_m & Y_m \end{bmatrix} \quad (6)$$

where x_i and y_i are presently available normalized loadings and X_i and Y_i, the desired normalized loadings, are functions of ϕ, the angle of rotation. The computational steps are as follows:

A. Calculation of NUM and DEN:

$$A = \sum_i (x_i + y_i)\ (x_i - y_i)$$

$$B = 2 \sum_i x_i\, y_i$$

$$C = \sum_i \left[(x_i + y_i)\ (x_i - y_i) + 2x_i\, y_i \right] \quad (7)$$

$$\left[(x_i + y_i)\ (x_i - y_i) - 2x_i\, y_i \right]$$

$$D = 4 \sum_i (x_i + y_i)\ (x_i - y_i)\ x_i\, y_i$$

$$NUM = D - 2AB/m$$

$$DEN = C - \left[(A + B)\ (A - B) \right] / m$$

B. Comparison of NUM and DEN:

The following four cases may arise:

NUM < DEN, go to B1 below.

NUM > DEN, go to B2 below.

(NUM + DEN) $\geq \epsilon$*, go to B3 below.

(NUM + DEN) $< \epsilon$, skip to the next rotation.

* ϵ is a small tolerance factor.

B1: $\tan 4\theta = |NUM|/|DEN|$ (8)

If $\tan 4\theta < \epsilon$ and

(i) DEN is positive, skip to the next rotation.

(ii) DEN is negative, set $\cos\phi = \sin\phi = (\sqrt{2})/2$ and go to E below.

If $\tan 4\theta \geq \epsilon$, calculate:

$$\cos 4\theta = 1/\sqrt{1 + \tan^2 4\theta} \quad (9)$$

$$\sin 4\theta = \tan 4\theta \cdot \cos 4\theta \quad (10)$$

and go to C below.

B2: $\operatorname{ctn} 4\theta = |NUM|/|DEN|$ (11)

If $\operatorname{ctn} 4\theta < \epsilon$, set $\cos 4\theta = 0$ and $\sin 4\theta = 1$. Go to C below.

If $\operatorname{ctn} 4\theta \geq \epsilon$, calculate:

$$\sin 4\theta = 1/\sqrt{1 + \operatorname{ctn}^2 4\theta} \quad (12)$$

$$\cos 4\theta = \operatorname{ctn} 4\theta \cdot \sin 4\theta \quad (13)$$

and go to C below.

B3: Set $\cos 4\theta = \sin 4\theta = (\sqrt{2})/2$ and go to C below.

C. Determining $\cos\theta$ and $\sin\theta$:

$$\cos 2\theta = \sqrt{(1 + \cos 4\theta)/2} \quad (14)$$

$$\sin 2\theta = \sin 4\theta / 2\cos 2\theta \quad (15)$$

$$\cos\theta = \sqrt{(1 + \cos 2\theta)/2} \quad (16)$$

$$\sin\theta = \sin 2\theta / 2\cos\theta \quad (17)$$

D. Determining $\cos\phi$ and $\sin\phi$:

D1: If DEN is positive, set

$$\cos\phi = \cos\theta \quad (18)$$

$$\sin\phi = \sin\theta \tag{19}$$

and go to (D2) below.

If DEN is negative, calculate

$$\cos\phi = \frac{\sqrt{2}}{2}\cos\theta + \frac{\sqrt{2}}{2}\sin\theta \tag{20}$$

$$\sin\phi = \left|\frac{\sqrt{2}}{2}\cos\theta - \frac{\sqrt{2}}{2}\sin\theta\right| \tag{21}$$

and go to (D2) below.

D2: If NUM is positive, set

$$\cos\phi = |\cos\phi| \tag{22}$$

$$\sin\phi = |\sin\phi| \tag{23}$$

and go to (E) below.

If NUM is negative, set

$$\cos\phi = |\cos\phi| \tag{24}$$

$$\sin\phi = -|\sin\phi| \tag{25}$$

E. Rotation:

$$X_i = x_i\cos\phi + y_i\sin\phi \tag{26}$$

$$Y_i = x_i\sin\phi + y_i\cos\phi \tag{27}$$

where $i = 1, 2, \ldots, m$

After one cycle of $k(k-1)/2$ rotations is completed, the subroutine goes back to calculate the variance for the factor matrix (equation 4).

Denormalization:

$$a_{ij} = b_{ij} \cdot h_i \tag{28}$$

where $i = 1, 2, \ldots, m$

$j = 1, 2, \ldots, k$

Check on communalities:

Final communalities $$f_i^2 = \sum_{j=1}^{k} a_{ij}^2 \tag{29}$$

Difference $$d_i = h_i^2 - f_i^2 \tag{30}$$

where $i = 1, 2, \ldots, m$

Subroutine VARMX

Purpose:

Perform orthogonal rotations of a factor matrix. This subroutine normally occurs in a sequence of calls to subroutines CORRE, EIGEN, TRACE, LOAD, VARMX in the performance of a factor analysis.

Usage:

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

Description of parameters:

- 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.

Remarks:

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

Subroutines and function subprograms required:

None.

Method:

Kaiser's varimax rotation as described in 'Computer Program for Varimax Rotation in Factor Analysis' by the same author, Educational and Psychological Measurement, Vol. XIX, No. 3, 1959.

```
      SUBROUTINE VARMX (M,K,A,NC,TV,H,F,D)                              VARMX  1
      DIMENSION A(1),TV(1),H(1),F(1),D(1)                               VARMX  2
C        INITIALIZATION                                                 VARMX  3
      EPS=0.00116                                                       VARMX  4
      TVLT=0.0                                                          VARMX  5
      LL=K-1                                                            VARMX  6
      NV=1                                                              VARMX  7
      NC=0                                                              VARMX  8
      FN=M                                                              VARMX  9
      FFN=FN*FN                                                         VARMX 10
      CONS=0.7071066                                                    VARMX 11
C        CALCULATE ORIGINAL COMMUNALITIES                               VARMX 12
      DO 110 I=1,M                                                      VARMX 13
      H(I)=0.0                                                          VARMX 14
      DO 110 J=1,K                                                      VARMX 15
      L=M*(J-1)+I                                                       VARMX 16
  110 H(I)=H(I)+A(L)*A(L)                                               VARMX 17
C        CALCULATE NORMALIZED FACTOR MATRIX                             VARMX 18
      DO 120 I=1,M                                                      VARMX 19
  115 H(I)= SQRT(H(I))                                                  VARMX 20
      DO 120 J=1,K                                                      VARMX 21
      L=M*(J-1)+I                                                       VARMX 22
  120 A(L)=A(L)/H(I)                                                    VARMX 23
      GO TO 132                                                         VARMX 24
C        CALCULATE VARIANCE FOR FACTOR MATRIX                           VARMX 25
  130 NV=NV+1                                                           VARMX 26
      TVLT=TV(NV-1)                                                     VARMX 27
  132 TV(NV)=0.0                                                        VARMX 28
      DO 150 J=1,K                                                      VARMX 29
      AA=0.0                                                            VARMX 30
      BB=0.0                                                            VARMX 31
      LB=M*(J-1)                                                        VARMX 32
      DO 140 I=1,M                                                      VARMX 33
      L=LB+I                                                            VARMX 34
      CC=A(L)*A(L)                                                      VARMX 35
      AA=AA+CC                                                          VARMX 36
  140 BB=BB+CC*CC                                                       VARMX 37
  150 TV(NV)=TV(NV)+(FN*BB-AA*AA)/FFN                                   VARMX 38
      IF(NV-51) 160, 430, 430                                           VARMX 39
C        PERFORM CONVERGENCE TEST                                       VARMX 40
  160 IF((TV(NV)-TVLT)-(1.E-7)) 170, 170, 190                           VARMX 41
  170 NC=NC+1                                                           VARMX 42
      IF(NC-3) 190, 190, 430                                            VARMX 43
C        ROTATION OF TWO FACTORS CONTINUES UP TO                        VARMX 44
C        THE STATEMENT 120.                                             VARMX 45
  190 DO 420 J=1,LL                                                     VARMX 46
      LI=M*(J-1)                                                        VARMX 47
      II=J+1                                                            VARMX 48
C        CALCULATE NUM AND DEN                                          VARMX 49
      DO 420 KI=II,K                                                    VARMX 50
      L2=M*(KI-1)                                                       VARMX 51
      AA=0.0                                                            VARMX 52
      BB=0.0                                                            VARMX 53
      CC=0.0                                                            VARMX 54
      DD=0.0                                                            VARMX 55
      DO 230 I=1,M                                                      VARMX 56
      L3=LI+I                                                           VARMX 57
      L4=L2+I                                                           VARMX 58
      U=(A(L3)+A(L4))*(A(L3)-A(L4))                                     VARMX 59
      T=A(L3)*A(L4)                                                     VARMX 60
      T=T+T                                                             VARMX 61
      CC=CC+(U+T)*(U-T)                                                 VARMX 62
      DD=DD+2.0*U*T                                                     VARMX 63
      AA=AA+U                                                           VARMX 64
  230 BB=BB+T                                                           VARMX 65
      T=DD-2.0*AA*BB/FN                                                 VARMX 66
      B=CC-(AA*AA-BB*BB)/FN                                             VARMX 67
C        COMPARISON OF NUM AND DEN                                      VARMX 68
      IF(T-B) 280, 240, 320                                             VARMX 69
  240 IF((T+B)-EPS) 420, 250, 250                                       VARMX 70
C        NUM + DEN IS GREATER THAN OR EQUAL TO THE                      VARMX 71
C        TOLERANCE FACTOR                                               VARMX 72
  250 COS4T=CONS                                                        VARMX 73
      SIN4T=CONS                                                        VARMX 74
      GO TO 350                                                         VARMX 75
C        NUM IS LESS THAN DEN                                           VARMX 76
  280 TAN4T= ABS(T)/ ABS(B)                                             VARMX 77
      IF(TAN4T-EPS) 300, 290, 290                                       VARMX 78
  290 COS4T=1.0/ SQRT(1.0+TAN4T*TAN4T)                                  VARMX 79
      SIN4T=TAN4T*COS4T                                                 VARMX 80
      GO TO 350                                                         VARMX 81
  300 IF(B) 310, 420, 420                                               VARMX 82
  310 SINP=CONS                                                         VARMX 83
      COSP=CONS                                                         VARMX 84
      GO TO 400                                                         VARMX 85
C        NUM IS GREATER THAN DEN                                        VARMX 86
  320 CTN4T= ABS(T/B)                                                   VARMX 87
      IF(CTN4T-EPS) 340, 330, 330                                       VARMX 88
  330 SIN4T=1.0/ SQRT(1.0+CTN4T*CTN4T)                                  VARMX 89
      COS4T=CTN4T*SIN4T                                                 VARMX 90
      GO TO 350                                                         VARMX 91
  340 COS4T=0.0                                                         VARMX 92
      SIN4T=1.0                                                         VARMX 93
C        DETERMINE COS THETA AND SIN THETA                              VARMX 94
  350 COS2T= SQRT((1.0+COS4T)/2.0)                                      VARMX 95
      SIN2T=SIN4T/(2.0*COS2T)                                           VARMX 96
  355 COST= SQRT((1.0+COS2T)/2.0)                                       VARMX 97
      SINT=SIN2T/(2.0*COST)                                             VARMX 98
C        DETERMINE COS PHI AND SIN PHI                                  VARMX 99
      IF(B) 370, 370, 360                                               VARMX100
  360 COSP=COST                                                         VARMX101
      SINP=SINT                                                         VARMX102
      GO TO 380                                                         VARMX103
  370 COSP=CONS*COST+CONS*SINT                                          VARMX104
  375 SINP= ABS(CONS*COST-CONS*SINT)                                    VARMX105
  380 IF(T) 390, 390, 400                                               VARMX106
  390 SINP=-SINP                                                        VARMX107
C        PERFORM ROTATION                                               VARMX108
  400 DO 410 I=1,M                                                      VARMX109
      L3=LI+I                                                           VARMX110
      L4=L2+I                                                           VARMX111
      AA=A(L3)*COSP+A(L4)*SINP                                          VARMX112
      A(L4)=-A(L3)*SINP+A(L4)*COSP                                      VARMX113
  410 A(L3)=AA                                                          VARMX114
  420 CONTINUE                                                          VARMX115
      GO TO 130                                                         VARMX116
C        DENORMALIZE VARIMAX LOADINGS                                   VARMX117
  430 DO 440 I=1,M                                                      VARMX118
      DO 440 J=1,K                                                      VARMX119
      L=M*(J-1)+I                                                       VARMX120
  440 A(L)=A(L)*H(I)                                                    VARMX121
C        CHECK ON COMMUNALITIES                                         VARMX122
      NC=NV-1                                                           VARMX123
      DO 450 I=1,M                                                      VARMX124
  450 H(I)=H(I)*H(I)                                                    VARMX125
      DO 470 I=1,M                                                      VARMX126
      F(I)=0.0                                                          VARMX127
      DO 460 J=1,K                                                      VARMX128
      L=M*(J-1)+I                                                       VARMX129
  460 F(I)=F(I)+A(L)*A(L)                                               VARMX130
  470 D(I)=H(I)-F(I)                                                    VARMX131
      RETURN                                                            VARMX132
      END                                                               VARMX133
```

Statistics - Time Series

AUTO

This subroutine calculates the autocovariances for lags 0, 1, 2, ..., (L-1), given a time series of observations A_1, A_2, ..., A_n and a number L.

$$R_j = \frac{1}{n-j+1} \sum_{i=1}^{n-j+1} (A_i - AVER)\ (A_{i+j-1} - AVER) \quad (1)$$

where $AVER = \frac{1}{n} \sum_{i=1}^{n} A_i$

n = number of observations in time series A.

j = 1, 2, 3, ..., L represent time lags 0, 1, 2, ..., (L-1).

Subroutine AUTO

Purpose:

To find autocovariances of series A for lags 0 to L-1.

Usage:

CALL AUTO (A, N, L, R)

Description of parameters:

- 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 is different from the length of A. N must be greater than L. Otherwise, R(1) is set to zero and this routine exits.

Subroutines and function subprograms required:

None.

Method:

The method described by R. B. Blackman and J. W. Tukey in The Measurement of Power Spectra, Dover Publications, Inc., New York, 1959.

```
      SUBROUTINE AUTO (A,N,L,R)                                      AUTO   1
      DIMENSION A(1),R(1)                                            AUTO   2
C        CALCULATE AVERAGE OF TIME SERIES A                          AUTO   3
      AVER=0.0                                                       AUTO   4
      IF(N-L) 50,50,100                                              AUTO M01
   50 R(1)=0.0                                                       AUTO M02
      RETURN                                                         AUTO M03
  100 DO 110 I=1,N                                                   AUTO M04
  110 AVER=AVER+A(I)                                                 AUTO   6
      FN=N                                                           AUTO   7
      AVER=AVER/FN                                                   AUTO   8
C        CALCULATE AUTOCOVARIANCES                                   AUTO  11
      DO 130 J=1,L                                                   AUTO  12
      NJ=N-J+1                                                       AUTO  13
      SUM=0.0                                                        AUTO  14
      DO 120 I=1,NJ                                                  AUTO  15
      IJ=I+J-1                                                       AUTO  16
  120 SUM=SUM+(A(I)-AVER)*(A(IJ)-AVER)                               AUTO M05
      FNJ=NJ                                                         AUTO  18
  130 R(J)=SUM/FNJ                                                   AUTO M06
      RETURN                                                         AUTO  20
      END                                                            AUTO  21
```

CROSS

This subroutine calculates the crosscovariances of series B lagging and leading A, given two time series $A_1, A_2, \ldots, A_n$ and $B_1, B_2, \ldots, B_n$ and given a number L.

(a) B lags A:

$$R_j = \frac{1}{n-j+1} \sum_{i=1}^{n-j+1} (A_i - AVERA)\,(B_{i+j-1} - AVERB) \qquad (1)$$

(b) B leads A:

$$S_j = \frac{1}{n-j+1} \sum_{i=1}^{n-j+1} (A_{i+j-1} - AVERA)\,(B_i - AVERB) \qquad (2)$$

where $AVERA = \frac{1}{n} \sum_{i=1}^{n} A_i$

$$AVERB = \frac{1}{n} \sum_{i=1}^{n} B_i$$

n = number of observations in each series.

j = 1, 2, ..., L represent time lags (or leads) of 0, 1, 2, ..., (L-1).

Subroutine CROSS

Purpose:

To find the crosscovariances of series A with series B (which leads and lags A).

Usage:

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

Description of parameters:

- 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 - Crosscovariance 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 not, R(1) and S(1) are set to zero and this routine exits.

Subroutines and function subprograms required:

None.

Method:

The method is described by R. B. Blackman and J. W. Tukey in The Measurement of Power Spectra, Dover Publications, Inc., New York, 1959.

```
      SUBROUTINE CROSS (A,B,N,L,R,S)                                    CROSS  1
      DIMENSION A(1),B(1),R(1),S(1)                                     CROSS  2
C        CALCULATE AVERAGES OF SERIES A AND B                           CROSS  3
      FN=N                                                              CROSS  4
      AVERA=0.0                                                         CROSS  5
      AVERB=0.0                                                         CROSS  6
      IF(N-L)50,50,100                                                  CROSSM01
   50 R(1)=0.0                                                          CROSSM02
      S(1)=0.0                                                          CROSSM03
      RETURN                                                            CROSSM04
  100 DO 110 I=1,N                                                      CROSSM05
      AVERA=AVERA+A(I)                                                  CROSS  8
  110 AVERB=AVERB+B(I)                                                  CROSS  9
      AVERA=AVERA/FN                                                    CROSS 10
      AVERB=AVERB/FN                                                    CROSS 11
C        CALCULATE CROSSCOVARIANCES OF SERIES A AND B                   CROSS 14
      DO 130 J=1,L                                                      CROSS 15
      NJ=N-J+1                                                          CROSS 16
      SUMR=0.0                                                          CROSS 17
      SUMS=0.0                                                          CROSS 18
      DO 120 I=1,NJ                                                     CROSS 19
      IJ=I+J-1                                                          CROSS 20
      SUMR=SUMR+(A(I)-AVERA)*(B(IJ)-AVERB)                              CROSSM06
  120 SUMS=SUMS+(A(IJ)-AVERA)*(B(I)-AVERB)                              CROSSM07
      FNJ=NJ                                                            CROSS 23
      R(J)=SUMR/FNJ                                                     CROSSM08
  130 S(J)=SUMS/FNJ                                                     CROSSM09
      RETURN                                                            CROSS 26
      END                                                               CROSS 27
```

SMO

This subroutine calculates the smoothed or filtered series, given a time series A_1, A_2, ..., A_n, a selection integer L, and a weighting series W_1, W_2, ..., W_m.

$$R_i = \sum_{j=1}^{m} A_p \cdot W_j \qquad (1)$$

where $p = j \cdot L - L + k$

$k = i - IL + 1$

$i = IL$ to IH

$$IL = \frac{L(m-1)}{2} + 1 \qquad (2)$$

$$IH = n - \frac{L(m-1)}{2} \qquad (3)$$

L = a given selection integer. For example, L = 4 applies weights to every 4th item of the time series.

m = number of weights. Must be an odd integer. (If m is an even integer, any fraction resulting from the calculation of $\frac{L(m-1)}{2}$ in (2) and (3) above will be truncated.

n = number of items in the time series.

From IL to IH elements of the vector R are filled with the smoothed series and other elements with zeros.

Subroutine SMO

Purpose:

To smooth or filter series A by weights W.

Usage:

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

Description of parameters:

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 12^{th} 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 of L*M.

Subroutines and function subprograms required:

None.

Method:

Refer to the article 'FORTRAN Subroutines for Time Series Analysis', by J. R. Healy and B. P. Bogert, Communications of ACM, V. 6, No. 1, Jan., 1963.

```
      SUBROUTINE SMO (A,N,W,M,L,R)                                      SMO   1
      DIMENSION A(1),W(1),R(1)                                          SMO   2
C        INITIALIZATION                                                 SMO   3
      DO 110 I=1,N                                                      SMO   4
  110 R(I)=0.0                                                          SMO   5
      IL=(L*(M-1))/2+1                                                  SMO   6
      IH=N-(L*(M-1))/2                                                  SMO   7
C        SMOOTH SERIES A BY WEIGHTS W                                   SMO   8
      DO 120 I=IL,IH                                                    SMO   9
      K=I-IL+1                                                          SMO  10
      DO 120 J=1,M                                                      SMO  11
      IP=(J*L)-L+K                                                      SMO  12
  120 R(I)=R(I)+A(IP)*W(J)                                              SMO  13
      RETURN                                                            SMO  14
      END                                                               SMO  15
```

EXSMO

This subroutine calculates a smoothed series S_1, S_2, ..., S_{NX}, given time series X_1, X_2, ..., X_{NX} and a smoothing constant α. Also, at the end of the computation, the coefficients A, B, and C are given for the expression $A + B(T) + C(T)^2/2$. This expression can be used to find estimates of the smoothed series a given number of time periods, T, ahead.

The subroutine has the following two stages for i = 1, 2, ..., NX, starting with A, B, and C either given by the user or provided automatically by the subroutine (see below).

(a) Find S_i for one period ahead

$$S_i = A + B + .5C \qquad (1)$$

(b) Update coefficients A, B, and C

$$A = X_i + (1-\alpha)^3 (S_i - X_i) \qquad (2)$$

$$B = B + C - 1.5(\alpha^2)(2-\alpha)(S_i - X_i) \qquad (3)$$

$$C = C - (\alpha^3)(S_i - X_i) \qquad (4)$$

where α = smoothing constant specified by the user

$(0.0 < \alpha < 1.0)$.

If coefficients A, B, and C are not all zero (0.0), take given values as initial values. However, if A = B = C = 0.0, generate initial values of A, B, and C as follows:

$$C = X_1 - 2X_2 + X_3 \qquad (5)$$

$$B = X_2 - X_1 - 1.5C \qquad (6)$$

$$A = X_1 - B - 0.5C \qquad (7)$$

Subroutine EXSMO

Purpose:

To find the triple exponential smoothed series S of the given series X.

Usage:

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

Description of parameters:

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 zero and less than one.

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, program will provide initial values. If at least one of A, B, C is not zero, program will take given values as initial values.

As output: A, B, C, contain latest, updated coefficients of prediction.

S - Output vector of length NX containing triple exponentially smoothed time series.

Remarks:

None.

Subroutines and function subprograms required:

None.

Method:

Refer to R. G. Brown, 'Smoothing, Forecasting and Prediction of Discrete Time Series', Prentice-Hall, N.J., 1963, pp. 140 to 144.

```
      SUBROUTINE EXSMO (X,NX,AL,A,B,C,S)                                EXSMO  1
      DIMENSION X(1),S(1)                                               EXSMO  2
C        IF A=B=C=0.0, GENERATE INITIAL VALUES OF A, B, AND C           EXSMO  3
      IF(A) 140, 110, 140                                               EXSMO  4
  110 IF(B) 140, 120, 140                                               EXSMO  5
  120 IF(C) 140, 130, 140                                               EXSMO  6
  130 L1=1                                                              EXSMOM01
      L2=2                                                              EXSMOM02
      L3=3                                                              EXSMOM03
      C=X(L1)-2.0*X(L2)+X(L3)                                           EXSMOM04
      B=X(L2)-X(L1)-1.5*C                                               EXSMOM05
      A=X(L1)-B-0.5*C                                                   EXSMOM06
  140 BE=1.0-AL                                                         EXSMO 10
      BECUB=BE*BE*BE                                                    EXSMO 11
      ALCUB=AL*AL*AL                                                    EXSMO 12
C        DO THE FOLLOWING FOR I=1 TO NX                                 EXSMO 13
      DO 150 I=1,NX                                                     EXSMO 14
C        FIND S(I) FOR ONE PERIOD AHEAD                                 EXSMO 15
      S(I)=A+B+0.5*C                                                    EXSMO 16
C        UPDATE COEFFICIENTS A, B, AND C                                EXSMO 17
      DIF=S(I)-X(I)                                                     EXSMO 18
      A=X(I)+BECUB*DIF                                                  EXSMO 19
      B=B+C-1.5*AL*AL*(2.0-AL)*DIF                                      EXSMO 20
  150 C=C-ALCUB*DIF                                                     EXSMO 21
      RETURN                                                            EXSMO 22
      END                                                               EXSMO 23
```

Statistics - Nonparametric

CHISQ

This subroutine calculates degrees of freedom and chi-square for a given contingency table A of observed frequencies with n rows (conditions) and m columns (groups). The degrees of freedom are:

$$\text{d.f.} = (n - 1)\ (m - 1) \tag{1}$$

If one or more cells have an expected value of less than 1, chi-square is computed and the error code is set to 1.

The following totals are computed:

$$T_i = \sum_{j=1}^{m} A_{ij};\ i = 1, 2, \ldots, n \text{ (row totals)} \tag{2}$$

$$T_j = \sum_{i=1}^{n} A_{ij};\ j = 1, 2, \ldots, m \text{ (column totals)} \tag{3}$$

$$GT = \sum_{i=1}^{n} T_i \text{ (grand total)} \tag{4}$$

Chi-square is obtained for two cases:

(a) for 2 x 2 table:

$$\chi^2 = \frac{GT\left(\left|A_{11}A_{22} - A_{12}A_{21}\right| - \frac{GT}{2}\right)^2}{(A_{11}+A_{12})(A_{21}+A_{22})(A_{11}+A_{21})(A_{12}+A_{22})} \tag{5}$$

(b) for other contingency tables:

$$\chi^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\left(A_{ij} - E_{ij}\right)^2}{E_{ij}} \tag{6}$$

where $E_{ij} = \dfrac{T_i T_j}{GT}$

$i = 1, 2, \ldots, n$

$j = 1, 2, \ldots, m$

Subroutine CHISQ

Purpose:

Compute chi-square from a contingency table.

Usage:

CALL CHISQ(A, N, M, CS, NDF, IERR, TR, TC)

Description of parameters:

- 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).
- IERR - Error code (output):
  - 0 - Normal case.
  - 1 - Expected value 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:

Chi-square is set to zero if either N or M is one (error code 3).

Subroutines and function subprograms required:

None.

Method:

Described in S. Siegel, 'Nonparametric Statistics for the Behavioral Sciences', McGraw-Hill, New York, 1956, Chapter 6 and Chapter 8.

```
      SUBROUTINE CHISQ(A,N,M,CS,NDF,IERR,TR,TC)                          CHISQ  1
      DIMENSION A(1),TR(1),TC(1)                                         CHISQ  2
      NM=N*M                                                             CHISQ  3
      IERR=0                                                             CHISQ  4
      CS=0.0                                                             CHISQ  5
C        FIND DEGREES OF FREEDOM                                         CHISQ  6
      NDF=(N-1)*(M-1)                                                    CHISQ  7
      IF(NDF) 5,5,10                                                     CHISQ  8
    5 IERR=3                                                             CHISQ  9
      RETURN                                                             CHISQ 10
C        COMPUTE TOTALS OF ROWS                                          CHISQ 25
   10 DO 90 I=1,N                                                        CHISQM01
      TR(I)=0.0                                                          CHISQ 27
      IJ=I-N                                                             CHISQ 28
      DO 90 J=1,M                                                        CHISQ 29
      IJ=IJ+N                                                            CHISQ 30
   90 TR(I)=TR(I)+A(IJ)                                                  CHISQ 31
C        COMPUTE TOTALS OF COLUMNS                                       CHISQ 32
      IJ=0                                                               CHISQ 33
      DO 100 J=1,M                                                       CHISQ 34
      TC(J)=0.0                                                          CHISQ 35
      DO 100 I=1,N                                                       CHISQ 36
      IJ=IJ+1                                                            CHISQ 37
  100 TC(J)=TC(J)+A(IJ)                                                  CHISQ 38
C        COMPUTE GRAND TOTAL                                             CHISQ 39
      GT=0.0                                                             CHISQ 40
      DO 110 I=1,N                                                       CHISQ 41
  110 GT=GT+TR(I)                                                        CHISQ 42
C        COMPUTE CHI SQUARE FOR 2 BY 2 TABLE (SPECIAL CASE)              CHISQ 43
      IF(NM-4) 130,120,130                                               CHISQ 44
  120 L1=1                                                               CHISQM04
      L2=2                                                               CHISQM05
      L3=3                                                               CHISQM06
      L4=4                                                               CHISQM07
      CS=GT*(ABS(A(L1)*A(L4)-A(L2)*A(L3))-GT/2.0)**2/(TC(L1)*TC(L2)      CHISQM08
     1*TR(L1)*TR(L2))                                                    CHISQM09
      RETURN                                                             CHISQ 47
C        COMPUTE CHI SQUARE FOR OTHER CONTINGENCY TABLES                 CHISQ 48
  130 IJ=0                                                               CHISQ 49
      DO 140 J=1,M                                                       CHISQ 50
      DO 140 I=1,N                                                       CHISQ 51
      IJ=IJ+1                                                            CHISQ 52
      E=TR(I)*TC(J)/GT                                                   CHISQ 53
      IF(E-1.0) 135, 140, 140                                            CHISQM02
  135 IERR=1                                                             CHISQM03
  140 CS=CS+(A(IJ)-E)*(A(IJ)-E)/E                                        CHISQ 54
      RETURN                                                             CHISQ 55
      END                                                                CHISQ 56
```

UTEST

This subroutine tests whether two independent groups are from the same population by means of the Mann-Whitney U-test, given an input vector A with smaller group preceding larger group. The scores for both groups are ranked together in ascending order. Tied observations are assigned the average of the tied ranks.

The sum of ranks in the larger group, R2, is calculated. The U statistic is then computed as follows:

$$U' = n_1 n_2 + \frac{n_2 (n_2 + 1)}{2} - R_2 \qquad (1)$$

where n_1 = number of cases in smaller group

n_2 = number of cases in larger group

$$U = n_1 n_2 - U'$$

$$\text{if } U' < U, \text{ set } U = U' \qquad (2)$$

A correction factor for ties is obtained:

$$T = \sum \frac{t^3 - t}{12} \qquad (3)$$

where t = number of observations tied for a given rank

The standard deviation is computed for two cases:

(a) if T = 0

$$s = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}} \qquad (4)$$

(b) if T > 0

$$s = \sqrt{\left(\frac{n_1 n_2}{N(N-1)}\right)\left(\frac{N^3 - N}{12} - T\right)} \qquad (5)$$

where N = total number of cases $(n_1 + n_2)$

The significance of U is then tested:

$$Z = \frac{U - \bar{X}}{s} \qquad (6)$$

where $\bar{X}$ = mean $= \frac{n_1 n_2}{2}$

Z is set to zero if n_2 is less than 20.

Subroutine UTEST

Purpose:

Test whether two independent groups are from the same population by means of Mann-Whitney U-test.

Usage:

CALL UTEST(A, R, N1, N2, U, Z)

Description of parameters:

- 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).

Remarks:

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

Subroutines and function subprograms required:

RANK
TIE

Method:

Described in S. Siegel, 'Nonparametric Statistics for the Behavioral Sciences', McGraw-Hill, New York, 1956, Chapter 6.

```
      SUBROUTINE UTEST(A,R,N1,N2,U,Z)                                   UTEST  1
      DIMENSION A(1),R(1)                                               UTEST  2
C        RANK SCORES FROM BOTH GROUP TOGETHER IN ASCENDING ORDER, AND   UTEST  3
C        ASSIGN TIED OBSERVATIONS AVERAGE OF TIED RANKS                 UTEST  4
      N=N1+N2                                                           UTEST  5
      CALL RANK(A,R,N)                                                  UTEST  6
      Z=0.0                                                             UTEST  7
C        SUM RANKS IN LARGER GROUP                                      UTEST  8
      R2=0.0                                                            UTEST  9
      NP=N1+1                                                           UTEST 10
      DO 10 I=NP,N                                                      UTEST 11
   10 R2=R2+R(I)                                                        UTEST 12
C        CALCULATE U                                                    UTEST 13
      FNX=N1*N2                                                         UTEST 14
      FN=N                                                              UTEST 15
      FN2=N2                                                            UTEST 16
      UP=FNX*FN2*((FN2+1.0)/2.0)-R2                                     UTEST 17
      U=FNX-UP                                                          UTEST 18
      IF(UP-U) 20,30,3)                                                 UTEST 19
   20 U=UP                                                              UTEST 20
C        TEST FOR N2 LESS THAN 20                                       UTEST 21
   30 IF(N2-20) 80,40,40                                                UTEST 22
C        COMPUTE STANDARD DEVIATION                                     UTEST 23
   40 KT=1                                                              UTEST 24
      CALL TIE(R,N,KT,TS)                                               UTEST 25
      IF(TS) 50,60,50                                                   UTEST 26
   50 S=SQRT((FNX/(FN*(FN-1.0)))*(((FN*FN*FN-FN)/12.0)-TS))             UTEST 27
      GO TO 70                                                          UTEST 28
   60 S=SQRT(FNX*(FN+1.0)/12.0)                                         UTEST 29
C        COMPUTE Z                                                      UTEST 30
   70 Z=(U-FNX*0.5)/S                                                   UTEST 31
   80 RETURN                                                            UTEST 32
      END                                                               UTEST 33
```

TWOAV

This subroutine determines the Friedman two-way analysis of variance statistic, given a matrix A with n rows (groups) and m columns (cases). Data in each group is ranked from 1 to m. Tied observations are assigned the average of the tied ranks.

The sum of ranks is calculated:

$$R_j = \sum_{i=1}^{n} A_{ij} \qquad (1)$$

Friedman's statistic is then computed:

$$\chi_r^2 = \frac{12}{nm(m+1)} \sum_{j=1}^{m} (R_j)^2 - 3n(m+1) \qquad (2)$$

The degrees of freedom are:

$$d.f = m - 1 \qquad (3)$$

Subroutine TWOAV

Purpose:

Test whether a number of samples are from the same population by the Friedman two-way analysis of variance test.

Usage:

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

Description of parameters:

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

Remarks:

None.

Subroutines and function subprograms required:

Rank.

Method:

Described in S. Siegel, 'Nonparametric Statistics for the Behavioral Sciences', McGraw-Hill, New York, 1956, Chapter 7.

```
      SUBROUTINE TWOAV (A,R,N,M,W,XR,NDF,NR)                          TWOAV  1
      DIMENSION A(1),R(1),W(1)                                        TWOAV  2
C        DETERMINE WHETHER DATA IS RANKED                             TWOAV  3
      IF(NR-1) 10, 30, 10                                             TWOAV  4
C        RANK DATA IN EACH GROUP AND ASSIGN TIED OBSERVATIONS AVERAGE TWOAV  5
C        OF TIED RANK                                                 TWOAV  6
   10 DO 20 I=1,N                                                     TWOAV  7
      IJ=I-N                                                          TWOAV  8
      IK=IJ                                                           TWOAV  9
      DO 15 J=1,M                                                     TWOAV 10
      IJ=IJ+N                                                         TWOAV 11
   15 W(J)=A(IJ)                                                      TWOAV 12
      CALL RANK (W,W(M+1),M)                                          TWOAV 13
      DO 20 J=1,M                                                     TWOAV 14
      IK=IK+N                                                         TWOAV 15
      IW=M+J                                                          TWOAV 16
   20 R(IK)=W(IW)                                                     TWOAV 17
      GO TO 35                                                        TWOAV 18
   30 NM=N*M                                                          TWOAV 19
      DO 32 I=1,NM                                                    TWOAV 20
   32 R(I)=A(I)                                                       TWOAV 21
C        CALCULATE SUM OF SQUARES OF SUMS OF RANKS                    TWOAV 22
   35 RTSQ=0.0                                                        TWOAV 23
      IR=0                                                            TWOAV 24
      DO 50 J=1,M                                                     TWOAV 25
      RT=0.0                                                          TWOAV 26
      DO 40 I=1,N                                                     TWOAV 27
      IR=IR+1                                                         TWOAV 28
   40 RT=RT+R(IR)                                                     TWOAV 29
   50 RTSQ=RTSQ+RT*RT                                                 TWOAV 30
C        CALCULATE FRIEDMAN TEST VALUE, XR                            TWOAV 31
      FNM=N*(M+1)                                                     TWOAV 32
      FM=M                                                            TWOAV 33
      XR=(12.0/(FM*FNM))*RTSQ-3.0*FNM                                 TWOAV 34
C        FIND DEGREES OF FREEDOM                                      TWOAV 35
      NDF=M-1                                                         TWOAV 36
      RETURN                                                          TWOAV 37
      END                                                             TWOAV 38
```

QTEST

This subroutine determines the Cochran Q-test statistic, given a matrix A of dichotomous data with n rows (sets) and m columns (groups).

Row and column totals are calculated:

$$L_i = \sum_{j=1}^{m} A_{ij} \text{ (row totals)} \qquad (1)$$

where i = 1, 2, ..., n

$$G_j = \sum_{i=1}^{n} A_{ij} \text{ (column totals)} \qquad (2)$$

where j = 1, 2, ..., m

The Cochran Q statistic is computed:

$$Q = \frac{(m-1)\left[m\sum_{j=1}^{m} G_j^2 - \left(\sum_{j=1}^{m} G_j\right)^2\right]}{m\sum_{i=1}^{n} L_i - \sum_{i=1}^{n} L_i^2} \qquad (3)$$

The degrees of freedom are:

$$d.f = m - 1 \qquad (4)$$

Subroutine QTEST

Purpose:

Test whether three or more matched groups of dichotomous data differ significantly by the Cochran Q-test.

Usage:

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

Description of parameters:

- 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 three or greater.

Subroutines and function subprograms required:

None.

Method:

Described in S. Siegel, 'Nonparametric Statistics for the Behavioral Sciences', McGraw-Hill, New York, 1956, Chapter 7.

```
      SUBROUTINE QTEST(A,N,M,Q,NDF)                                     QTEST  1
      DIMENSION A(1)                                                    QTEST  2
C        COMPUTE SUM OF SQUARES OF ROW TOTALS, RSQ, AND GRAND TOTAL OF  QTEST  3
C        ALL ELEMENTS, GD                                               QTEST  4
      RSQ=0.0                                                           QTEST  5
      GD=0.0                                                            QTEST  6
      DO 20 I=1,N                                                       QTEST  7
      TR=0.0                                                            QTEST  8
      IJ=I-N                                                            QTEST  9
      DO 10 J=1,M                                                       QTEST 10
      IJ=IJ+N                                                           QTEST 11
   10 TR=TR+A(IJ)                                                       QTEST 12
      GD=GD+TR                                                          QTEST 13
   20 RSQ=RSQ+TR*TR                                                     QTEST 14
C        COMPUTE SUM OF SQUARES OF COLUMN TOTALS, CSQ                   QTEST 15
      CSQ=0.0                                                           QTEST 16
      IJ=0                                                              QTEST 17
      DO 40 J=1,M                                                       QTEST 18
      TC=0.0                                                            QTEST 19
      DO 30 I=1,N                                                       QTEST 20
      IJ=IJ+1                                                           QTEST 21
   30 TC=TC+A(IJ)                                                       QTEST 22
   40 CSQ=CSQ+TC*TC                                                     QTEST 23
C        COMPUTE COCHRAN Q TEST VALUE                                   QTEST 24
      FM=M                                                              QTEST 25
      Q=(FM-1.0)*(FM*CSQ-GD*GD)/(FM*GD-RSQ)                             QTEST 26
C        FIND DEGREES OF FREEDOM                                        QTEST 27
      NDF=M-1                                                           QTEST 28
      RETURN                                                            QTEST 29
      END                                                               QTEST 30
```

SRANK

This subroutine measures the correlation between two variables by means of the Spearman rank correlation coefficient, given two vectors of n observations for the variables.

The observations on each variable are ranked from 1 to n. Tied observations are assigned the average of the tied ranks.

The sum of squares of rank differences is calculated:

$$D = \sum_{i=1}^{n} (A_i - B_i)^2 \tag{1}$$

where A_i = first ranked vector

B_i = second ranked vector

n = number of ranks

A correction factor for ties is obtained:

$$T_a = \sum \frac{t^3 - t}{12} \text{ over variable A}$$

$$T_b = \sum \frac{t^3 - t}{12} \text{ over variable B} \tag{2}$$

where t = number of observations tied for a given rank

The Spearman rank correlation coefficient is then computed for the following two cases:

(a) if T_a and T_b are zero,

$$r_s = 1 - \frac{6D}{n^3 - n} \tag{3}$$

(b) if T_a and/or T_b are not zero,

$$r_s = \frac{X + Y - D}{2\sqrt{XY}} \tag{4}$$

where $X = \frac{n^3 - n}{12} - T_a$ (5)

$$Y = \frac{n^3 - n}{12} - T_b \tag{6}$$

The statistic used to measure the significance of r_s is:

$$t = r_s \sqrt{\frac{n - 2}{1 - r_s^2}} \tag{7}$$

The degrees of freedom are:

$$d.f. = n - 2 \tag{8}$$

Subroutine SRANK

Purpose:

Test correlation between two variables by means of Spearman rank correlation coefficient.

Usage:

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

Description of parameters:

- 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:

T is set to zero if N is less than ten.

Subroutines and function subprograms required:

RANK
TIE

Method:

Described in S. Siegel, 'Nonparametric Statistics for the Behavioral Sciences', McGraw-Hill, New York, 1956, Chapter 9.

```
      SUBROUTINE SRANK(A,B,R,N,RS,T,NDF,NR)                          SRANK  1
      DIMENSION A(1),B(1),R(1)                                       SRANK  2
      D=N                                                            SRANKM01
      FNNN=D**3-D                                                    SRANKM02
C        DETERMINE WHETHER DATA IS RANKED                            SRANK  4
      IF(NR-1) 5, 10, 5                                              SRANK  5
C        RANK DATA IN A AND B VECTORS AND ASSIGN TIED OBSERVATIONS    SRANK  6
C        AVERAGE OF TIED RANKS                                       SRANK  7
    5 CALL RANK (A,R,N)                                              SRANK  8
```

```
      CALL RANK (B,R(N+1),N)                                          SRANK  9
      GO TO 40                                                        SRANK 10
C        MOVE RANKED DATA TO R VECTOR                                 SRANK 11
   10 DO 20 I=1,N                                                     SRANK 12
   20 R(I)=A(I)                                                       SRANK 13
      DO 30 I=1,N                                                     SRANK 14
      J=I+N                                                           SRANK 15
   30 R(J)=B(I)                                                       SRANK 16
C        COMPUTE SUM OF SQUARES OF RANK DIFFERENCES                   SRANK 17
   40 D=0.0                                                           SRANK 18
      DO 50 I=1,N                                                     SRANK 19
      J=I+N                                                           SRANK 20
   50 D=D+(R(I)-R(J))*(R(I)-R(J))                                     SRANK 21
C        COMPUTE TIED SCORE INDEX                                     SRANK 22
      KT=1                                                            SRANK 23
      CALL TIE (R,N,KT,TSA)                                           SRANK 24
      CALL TIE (R(N+1),N,KT,TSB)                                      SRANK 25
C        COMPUTE SPEARMAN RANK CORRELATION COEFFICIENT                SRANK 26
      IF(TSA) 60,55,60                                                SRANK 27
   55 IF(TSB) 60,57,60                                                SRANK 28
   57 RS=1.0-6.0*D/FNNN                                               SRANK 29
      GO TO 70                                                        SRANK 30
   60 X=FNNN/12.0-TSA                                                 SRANK 31
      Y=X+TSA-TSB                                                     SRANK 32
      RS=(X+Y-D)/(2.0*(SQRT(X*Y)))                                    SRANK 33
C        COMPUTE T AND DEGREES OF FREEDOM IF N IS 10 OR LARGER        SRANK 34
      T=0.0                                                           SRANK 35
   70 IF(N-10) 80,75,75                                               SRANK 36
   75 T=RS*SQRT(FLOAT(N-2)/(1.0-RS*RS))                               SRANK 37
   80 NDF=N-2                                                         SRANK 38
      RETURN                                                          SRANK 39
      END                                                             SRANK 40
```

KRANK

The subroutine computes the Kendall rank correlation coefficient, given two vectors of n observations for two variables, A and B. The observations on each variable are ranked from 1 to n. Tied observations are assigned the average of the tied ranks. Ranks are sorted in sequence of variable A.

A correction factor for ties is obtained:

$$T_a = \sum \frac{t(t-1)}{2} \text{ for variable A}$$

$$T_b = \sum \frac{t(t-1)}{2} \text{ for variable B} \qquad (1)$$

where t = number of observations tied for a given rank

The Kendall rank correlation coefficient is then computed for the following two cases:

(a) if T_a and T_b are zero,

$$\tau = \frac{S}{\frac{1}{2}n(n-1)} \qquad (2)$$

where n = number of ranks

S = total score calculated for ranks in variable B as follows: selecting each rank in turn, add 1 for each larger rank to its right, subtract 1 for each smaller rank to its right.

(b) if T_a and/or T_b are not zero,

$$\tau = \frac{S}{\sqrt{\frac{1}{2}n(n-1) - T_a}\sqrt{\frac{1}{2}n(n-1) - T_b}} \qquad (3)$$

The standard deviation is calculated:

$$s = \sqrt{\frac{2(2n+5)}{9n(n-1)}} \qquad (4)$$

The significance of τ can be measured by:

$$z = \frac{\tau}{s} \qquad (5)$$

Subroutine KRANK

Purpose:

Test correlation between two variables by means of Kendall rank correlation coefficient.

Usage:

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

Description of parameters:

- 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 ten.

Subroutines and function subprograms required:

RANK
TIE

Method:

Described in S. Siegel, 'Nonparametric Statistics for the Behavioral Sciences', McGraw-Hill, New York, 1956, Chapter 9.

```
      SUBROUTINE KRANK(A,B,R,N,TAU,SD,Z,NR)                              KRANK  1
      DIMENSION A(1),B(1),R(1)                                           KRANK  2
      SD=0.0                                                             KRANK  3
      Z=0.0                                                              KRANK  4
      FN=N                                                               KRANK  5
      FN1=N*(N-1)                                                        KRANK  6
C        DETERMINE WHETHER DATA IS RANKED                                KRANK  7
      IF(NR-1) 5, 10, 5                                                  KRANK  8
C        RANK DATA IN A AND B VECTORS AND ASSIGN TIED OBSERVATIONS        KRANK  9
C        AVERAGE OF TIED RANKS                                           KRANK 10
    5 CALL RANK (A,R,N)                                                  KRANK 11
      CALL RANK (B,R(N+1),N)                                             KRANK 12
      GO TO 40                                                           KRANK 13
C        MOVE RANKED DATA TO R VECTOR                                    KRANK 14
   10 DO 20 I=1,N                                                        KRANK 15
   20 R(I)=A(I)                                                          KRANK 16
      DO 30 I=1,N                                                        KRANK 17
      J=I+N                                                              KRANK 18
   30 R(J)=B(I)                                                          KRANK 19
C        SORT RANK VECTOR R IN SEQUENCE OF VARIABLE A                    KRANK 20
   40 ISORT=0                                                            KRANK 21
      DO 50 I=2,N                                                        KRANK 22
      IF(R(I)-R(I-1)) 45,50,50                                           KRANK 23
   45 ISORT=ISORT+1                                                      KRANK 24
      RSAVE=R(I)                                                         KRANK 25
      R(I)=R(I-1)                                                        KRANK 26
      R(I-1)=RSAVE                                                       KRANK 27
      I2=I+N                                                             KRANK 28
      SAVER=R(I2)                                                        KRANK 29
      R(I2)=R(I2-1)                                                      KRANK 30
      R(I2-1)=SAVER                                                      KRANK 31
   50 CONTINUE                                                           KRANK 32
      IF(ISORT) 40,55,40                                                 KRANK 33
C        COMPUTE S ON VARIABLE B. STARTING WITH THE FIRST RANK, ADD 1    KRANK 34
C        TO S FOR EACH LARGER RANK TO ITS RIGHT AND SUBTRACT 1 FOR EACH  KRANK 35
C        SMALLER RANK.  REPEAT FOR ALL RANKS.                            KRANK 36
   55 S=0.0                                                              KRANK 37
      NM=N-1                                                             KRANK 38
      DO 60 I=1,NM                                                       KRANK 39
      J=N+I                                                              KRANK 40
      DO 60 L=I,N                                                        KRANK 41
      K=N+L                                                              KRANK 42
      IF(R(K)-R(J)) 56,60,57                                             KRANK 43
   56 S=S-1.0                                                            KRANK 44
      GO TO 60                                                           KRANK 45
   57 S=S+1.0                                                            KRANK 46
   60 CONTINUE                                                           KRANK 47
C        COMPUTE TIED SCORE INDEX FOR BOTH VARIABLES                     KRANK 48
      KT=2                                                               KRANK 49
      CALL TIE(R,N,KT,TA)                                                KRANK 50
      CALL TIE(R(N+1),N,KT,TB)                                           KRANK 51
C        COMPUTE TAU                                                     KRANK 52
      IF(TA) 70,65,70                                                    KRANK 53
   65 IF(TB) 70,67,70                                                    KRANK 54
   67 TAU=S/(0.5*FN1)                                                    KRANK 55
      GO TO 90                                                           KRANK 56
   70 TAU=S/((SQRT(0.5*FN1-TA))*(SQRT(0.5*FN1-TB)))                      KRANK 57
C        COMPUTE STANDARD DEVIATION AND Z IF N IS 10 OR LARGER           KRANK 58
   80 IF(N-10) 90,85,85                                                  KRANK 59
   85 SD=(SQRT((2.0*(FN+FN+5.0))/(9.0*FN1)))                             KRANK 60
      Z=TAU/SD                                                           KRANK 61
   90 RETURN                                                             KRANK 62
      END                                                                KRANK 63
```

WTEST

This subroutine computes the Kendall coefficient of concordance, given a matrix A of n rows (variables) and m columns (cases). The observations on all variables are ranked from 1 to m. Tied observations are assigned the average of the tied ranks.

A correction factor for ties is obtained:

$$T = \sum_{i=1}^{n} \frac{t^3 - t}{12} \qquad (1)$$

where t = number of observations tied for a given rank

Sums of ranks are calculated:

$$Y_j = \sum_{i=1}^{n} R_{ij} \qquad (2)$$

where j = 1, 2, ..., m

From these, the mean of sums of ranks is found:

$$\bar{R} = \frac{\sum_{j=1}^{m} Y_j}{m} \qquad (3)$$

The sum of squares of deviations is derived:

$$s = \sum_{j=1}^{m} (Y_j - \bar{R})^2 \qquad (4)$$

The Kendall coefficient of concordance is then computed:

$$W = \frac{s}{\frac{1}{12}n^2 (m^3 - m) - nT} \qquad (5)$$

For m larger than 7, chi-square is:

$$\chi^2 = n(m-1)W \qquad (6)$$

The degrees of freedom are:

$$d.f. = n - 1 \qquad (7)$$

Subroutine WTEST

Purpose:

Test degree of association among a number of variables by the Kendall coefficient of concordance.

Usage:

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

Description of parameters:

- 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:

Chi-square is set to zero if M is 7 or smaller.

Subroutines and function subprograms required:

RANK
TIE

Method:

Described in S. Siegel, 'Nonparametric Statistics for the Behavioral Sciences', McGraw-Hill, New York, 1956, Chapter 9.

```
      SUBROUTINE WTEST (A,R,N,M,WA,W,CS,NDF,NR)                         WTEST  1
      DIMENSION A(1),R(1),WA(1)                                         WTEST  2
      FM=M                                                              WTEST  3
      FN=N                                                              WTEST  4
C        DETERMINE WHETHER DATA IS RANKED                               WTEST  5
C        RANK DATA FOR ALL VARIABLES ASSIGNING TIED OBSERVATIONS AVERAGEWTEST  6
C        OF TIED RANKS AND COMPUTE CORRECTION FOR TIED SCORES           WTEST  7
      T=0.0                                                             WTEST  8
      KT=1                                                              WTEST  9
      DO 20 I=1,N                                                       WTEST 10
      IJ=I-N                                                            WTEST 11
      IK=IJ                                                             WTEST 12
      IF(NR-1) 5,2,5                                                    WTEST 13
    2 DO 3 J=1,M                                                        WTEST 14
      IJ=IJ+N                                                           WTEST 15
      K=M+J                                                             WTEST 16
    3 WA(K)=A(IJ)                                                       WTEST 17
      GO TO 15                                                          WTEST 18
    5 DO 10 J=1,M                                                       WTEST 19
      IJ=IJ+N                                                           WTEST 20
   10 WA(J)=A(IJ)                                                       WTEST 21
      CALL RANK(WA,WA(M+1),M)                                           WTEST 22
   15 CALL TIE(WA(M+1),M,KT,TI)                                         WTEST 23
      T=T+TI                                                            WTEST 24
      DO 20 J=1,M                                                       WTEST 25
      IK=IK+N                                                           WTEST 26
      IW=M+J                                                            WTEST 27
   20 R(IK)=WA(IW)                                                      WTEST 28
C        CALCULATE VECTOR OF SUMS OF RANKS                              WTEST 29
      IR=0                                                              WTEST 30
      DO 40 J=1,M                                                       WTEST 31
      WA(J)=0.0                                                         WTEST 32
      DO 40 I=1,N                                                       WTEST 33
      IR=IR+1                                                           WTEST 34
   40 WA(J)=WA(J)+R(IR)                                                 WTEST 35
C        COMPUTE MEAN OF SUMS OF RANKS                                  WTEST 36
      SM=0.0                                                            WTEST 37
      DO 50 J=1,M                                                       WTEST 38
   50 SM=SM+WA(J)                                                       WTEST 39
      SM=SM/FM                                                          WTEST 40
C        COMPUTE SUM OF SQUARES OF DEVIATIONS                           WTEST 41
      S=0.0                                                             WTEST 42
      DO 60 J=1,M                                                       WTEST 43
   60 S=S+(WA(J)-SM)*(WA(J)-SM)                                         WTEST 44
C        COMPUTE W                                                      WTEST 45
      W=S/(((FN*FN)*(FM*FM*FM-FM)/12.0)-FN*T)                           WTEST 46
C        COMPUTE DEGREES OF FREEDOM AND CHI-SQUARE IF M IS OVER 7       WTEST 47
      CS=0.0                                                            WTEST 48
      NDF=0                                                             WTEST 49
      IF(M-7) 70,70,65                                                  WTEST 50
   65 CS=FN*(FM-1.0)*W                                                  WTEST 51
      NDF=M-1                                                           WTEST 52
   70 RETURN                                                            WTEST 53
      END                                                               WTEST 54
```

RANK

Purpose:
Rank a vector of values.

Usage:
CALL RANK(A, R, N)

Description of parameters:
- 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.

Remarks:
None.

Subroutines and function subprograms required:
None.

Method:
Vector is searched for successively larger elements. If ties occur, they are located and their rank value computed. For example, if two values are tied for sixth rank, they are assigned a rank of 6.5 (=(6+7)/2).

```
      SUBROUTINE RANK(A,R,N)                                          RANK   1
      DIMENSION A(1),R(1)                                             RANK   2
C        INITIALIZATION                                               RANK   3
      DO 10 I=1,N                                                     RANK   4
   10 R(I)=0.0                                                        RANK   5
C        FIND RANK OF DATA                                            RANK   6
      DO 100 I=1,N                                                    RANK   7
C        TEST WHETHER DATA POINT IS ALREADY RANKED                    RANK   8
      IF(R(I)) 20, 20, 100                                            RANK   9
C        DATA POINT TO BE RANKED                                      RANK  10
   20 SMALL=0.0                                                       RANK  11
      EQUAL=0.0                                                       RANK  12
      X=A(I)                                                          RANK  13
      DO 50 J=1,N                                                     RANK  14
      IF(A(J)-X) 30, 40, 50                                           RANK  15
C        COUNT NUMBER OF DATA POINTS WHICH ARE SMALLER                RANK  16
   30 SMALL=SMALL+1.0                                                 RANK  17
      GO TO 50                                                        RANK  18
C        COUNT NUMBER OF DATA POINTS WHICH ARE EQUAL                  RANK  19
   40 EQUAL=EQUAL+1.0                                                 RANK  20
      R(J)=-1.0                                                       RANK  21
   50 CONTINUE                                                        RANK  22
C        TEST FOR TIE                                                 RANK  23
      IF(EQUAL-1.0) 60, 60, 70                                        RANK  24
C        STORE RANK OF DATA POINT WHERE NO TIE                        RANK  25
   60 R(I)=SMALL+1.0                                                  RANK  26
      GO TO 100                                                       RANK  27
C        CALCULATE RANK OF TIED DATA POINTS                           RANK  28
   70 P=SMALL+(EQUAL+1.0)/2.0                                         RANK M01
      DO 90 J=1,N                                                     RANK  30
      IF(R(J)+1.0) 90, 80, 90                                         RANK  31
   80 R(J)=P                                                          RANK  32
   90 CONTINUE                                                        RANK  33
  100 CONTINUE                                                        RANK  34
      RETURN                                                          RANK  35
      END                                                             RANK  36
```

TIE

Purpose:
Calculate correction factor due to ties.

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

Description of parameters:
- 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.

Remarks:
None.

Subroutines and function subprograms required:
None.

Method:
Vector is searched for successively larger ranks. Ties are counted and correction factor 1 or 2 summed.

```
      SUBROUTINE TIE(R,N,KT,T)                                        TIE    1
      DIMENSION R(1)                                                  TIE    2
C        INITIALIZATION                                               TIE    3
      T=0.0                                                           TIE    4
      Y=0.0                                                           TIE    5
    5 X=1.0E38                                                        TIE    6
      IND=0                                                           TIE    7
C        FIND NEXT LARGEST RANK                                       TIE    8
      DO 30 I=1,N                                                     TIE    9
      IF(R(I)-Y) 30,30,10                                             TIE   10
   10 IF(R(I)-X) 20,30,30                                             TIE   11
   20 X=R(I)                                                          TIE   12
      IND=IND+1                                                       TIE   13
   30 CONTINUE                                                        TIE   14
C        IF ALL RANKS HAVE BEEN TESTED, RETURN                        TIE   15
      IF(IND) 90,90,40                                                TIE   16
   40 Y=X                                                             TIE   17
      CT=0.0                                                          TIE   18
C        COUNT TIES                                                   TIE   19
      DO 60 I=1,N                                                     TIE   20
      IF(R(I)-X) 60,50,60                                             TIE   21
   50 CT=CT+1.0                                                       TIE   22
   60 CONTINUE                                                        TIE   23
C        CALCULATE CORRECTION FACTOR                                  TIE   24
      IF(CT) 70,5,70                                                  TIE   25
   70 IF(KT-1) 75,80,75                                               TIE   26
   75 T=T+CT*(CT-1.)/2.0                                              TIE   27
      GO TO 5                                                         TIE   28
   80 T=T+(CT*CT*CT-CT)/12.0                                          TIE   29
      GO TO 5                                                         TIE   30
   90 RETURN                                                          TIE   31
      END                                                             TIE   32
```

RANDU

Purpose:

Computes uniformly distributed random floating point numbers between 0 and 1.0 and integers in the range 0 to 2**15.

Usage:

CALL RANDU(IX, IY, YFL)

Description of parameters:

IX - For the first entry this must contain any odd positive integer less than 32,768. After the first entry, IX should be the previous value of IY computed by this subroutine.

IY - A resultant integer random number required for the next entry to this subroutine. The range of this number is from zero to 2**15.

YFL - The resultant uniformly distributed, floating point, random number in the range 0 to 1.0.

Remarks:

This subroutine is specific to the IBM 1130. This subroutine should not repeat its cycle in less than 2 to the 13th entries.

Note: If random bits are needed, the high order bits of IY should be chosen.

Subroutines and function subprograms required:

None.

Method:

Power residue method discussed in IBM manual Random Number Generation and Testing (C20-8011).

```
      SUBROUTINE RANDU(IX,IY,YFL)                                RANDU  1
      IY=IX*899                                                  RANDU  2
      IF(IY)5,6,6                                                RANDU  3
    5 IY=IY+32767+1                                              RANDU  4
    6 YFL=IY                                                     RANDU  5
      YFL=YFL/32767.                                             RANDU  6
      RETURN                                                     RANDU  7
      END                                                        RANDU  8
```

GAUSS

This subroutine computes a normally distributed random number with a given mean and standard deviation.

An approximation to normally distributed random numbers Y can be found from a sequence of uniform random numbers* using the formula:

$$Y = \frac{\sum_{i=1}^{K} X_i - \frac{K}{2}}{\sqrt{K/12}} \qquad (1)$$

where X_i is a uniformly distributed random number, $0 < X_i < 1$

K is the number of values X_i to be used

Y approaches a true normal distribution asympototically as K approaches infinity. For this subroutine, K was chosen as 12 to reduce execution time. Equation (1) thus becomes:

$$Y = \sum_{i=1}^{12} X_i - 6.0$$

The adjustment for the required mean and standard deviation is then

$$Y' = Y * S + AM \qquad (2)$$

where Y' is the required normally distributed random number

S is the required standard deviation

AM is the required mean

* R. W. Hamming, Numerical Methods for Scientists and Engineers, McGraw-Hill, N.Y., 1962, pages 34 and 389.

Subroutine GAUSS

Purpose:
Computes a normally distributed random number with a given mean and standard deviation.

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

Description of parameters:
- IX - IX must contain an odd positive integer less than 32,768. Thereafter it 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:
This subroutine uses RANDU which is machine specific.

Subroutines and function subprograms required:
RANDU

Method:
Uses 12 uniform random numbers to compute normal random numbers by central limit theorem. The result is then adjusted to match the given mean and standard deviation. The uniform random numbers computed within the subroutine are found by the power residue method.

```
      SUBROUTINE GAUSS(IX,S,AM,V)                                       GAUSS  1
      A=0.0                                                             GAUSS  2
      DO 50 I=1,12                                                      GAUSS  3
      CALL RANDU(IX,IY,Y)                                               GAUSS  4
      IX=IY                                                             GAUSS  5
   50 A=A+Y                                                             GAUSS  6
      V=(A-6.0)*S+AM                                                    GAUSS  7
      RETURN                                                            GAUSS  8
      END                                                               GAUSS  9
```

Mathematics - Special Matrix Operations

MINV

Purpose:
Invert a matrix.

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

Description of parameters:
- 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.

Subroutines and function subprograms required:
None.

Method:
The standard Gauss-Jordan method is used. The determinant is also calculated. A determinant with absolute value less than 10**(-20) indicates singularity. The user may wish to change this.

```
      SUBROUTINE MINV(A,N,D,L,M)                                        MINV   1
      DIMENSION A(1),L(1),M(1)                                          MINV   2
C        SEARCH FOR LARGEST ELEMENT                                     MINV   3
      D=1.0                                                             MINV   4
      NK=-N                                                             MINV   5
      DO 80 K=1,N                                                       MINV   6
      NK=NK+N                                                           MINV   7
      L(K)=K                                                            MINV   8
      M(K)=K                                                            MINV   9
      KK=NK+K                                                           MINV  10
      BIGA=A(KK)                                                        MINV  11
      DO 20 J=K,N                                                       MINV  12
      IZ=N*(J-1)                                                        MINV  13
      DO 20 I=K,N                                                       MINV  14
      IJ=IZ+I                                                           MINV  15
   10 IF( ABS(BIGA)- ABS(A(IJ))) 15,20,20                               MINV  16
   15 BIGA=A(IJ)                                                        MINV  17
      L(K)=I                                                            MINV  18
      M(K)=J                                                            MINV  19
   20 CONTINUE                                                          MINV  20
C        INTERCHANGE ROWS                                               MINV  21
      J=L(K)                                                            MINV  22
      IF(J-K) 35,35,25                                                  MINV  23
   25 KI=K-N                                                            MINV  24
      DO 30 I=1,N                                                       MINV  25
      KI=KI+N                                                           MINV  26
      HOLD=-A(KI)                                                       MINV  27
      JI=KI-K+J                                                         MINV  28
      A(KI)=A(JI)                                                       MINV  29
   30 A(JI) =HOLD                                                       MINV  30
C        INTERCHANGE COLUMNS                                            MINV  31
   35 I=M(K)                                                            MINV  32
      IF(I-K) 45,45,38                                                  MINV  33
   38 JP=N*(I-1)                                                        MINV  34
      DO 40 J=1,N                                                       MINV  35
      JK=NK+J                                                           MINV  36
      JI=JP+J                                                           MINV  37
      HOLD=-A(JK)                                                       MINV  38
      A(JK)=A(JI)                                                       MINV  39
   40 A(JI) =HOLD                                                       MINV  40
C        DIVIDE COLUMN BY MINUS PIVOT (VALUE OF PIVOT ELEMENT IS        MINV  41
C        CONTAINED IN BIGA)                                             MINV  42
   45 IF(ABS(BIGA)-1.E-20)46,46,48                                      MINV M03
   46 D=0.0                                                             MINV  44
      RETURN                                                            MINV  45
   48 DO 55 I=1,N                                                       MINV  46
      IF(I-K) 50,55,50                                                  MINV  47
   50 IK=NK+I                                                           MINV  48
      A(IK)=A(IK)/(-BIGA)                                               MINV  49
   55 CONTINUE                                                          MINV  50
C        REDUCE MATRIX                                                  MINV  51
      DO 65 I=1,N                                                       MINV  52
      IK=NK+I                                                           MINV  53
```

```
      HOLD=A(IK)                                                    MINV M01
      IJ=I-N                                                        MINV  54
      DO 65 J=1,N                                                   MINV  55
      IJ=IJ+N                                                       MINV  56
      IF(I-K) 60,65,60                                              MINV  57
   60 IF(J-K) 62,65,62                                              MINV  58
   62 KJ=IJ-I+K                                                     MINV  59
      A(IJ)=HOLD*A(KJ)+A(IJ)                                        MINV M02
   65 CONTINUE                                                      MINV  61
C        DIVIDE ROW BY PIVOT                                        MINV  62
      KJ=K-N                                                        MINV  63
      DO 75 J=1,N                                                   MINV  64
      KJ=KJ+N                                                       MINV  65
      IF(J-K) 70,75,70                                              MINV  66
   70 A(KJ)=A(KJ)/BIGA                                              MINV  67
   75 CONTINUE                                                      MINV  68
C        PRODUCT OF PIVOTS                                          MINV  69
      D=D*BIGA                                                      MINV  70
C        REPLACE PIVOT BY RECIPROCAL                                MINV  71
      A(KK)=1.0/BIGA                                                MINV  72
   80 CONTINUE                                                      MINV  73
C        FINAL ROW AND COLUMN INTERCHANGE                           MINV  74
      K=N                                                           MINV  75
  100 K=(K-1)                                                       MINV  76
      IF(K) 150,150,105                                             MINV  77
  105 I=L(K)                                                        MINV  78
      IF(I-K) 120,120,108                                           MINV  79
  108 JQ=N*(K-1)                                                    MINV  80
      JR=N*(I-1)                                                    MINV  81
      DO 110 J=1,N                                                  MINV  82
      JK=JQ+J                                                       MINV  83
      HOLD=A(JK)                                                    MINV  84
      JI=JR+J                                                       MINV  85
      A(JK)=-A(JI)                                                  MINV  86
  110 A(JI) =HOLD                                                   MINV  87
  120 J=M(K)                                                        MINV  88
      IF(J-K) 100,100,125                                           MINV  89
  125 KI=K-N                                                        MINV  90
      DO 130 I=1,N                                                  MINV  91
      KI=KI+N                                                       MINV  92
      HOLD=A(KI)                                                    MINV  93
      JI=KI-K+J                                                     MINV  94
      A(KI)=-A(JI)                                                  MINV  95
  130 A(JI) =HOLD                                                   MINV  96
      GO TO 100                                                     MINV  97
  150 RETURN                                                        MINV  98
      END                                                           MINV  99
```

EIGEN

This subroutine computes the eigenvalues and eigenvectors of a real symmetric matrix.

Given a symmetric matrix A of order N, eigenvalues are to be developed in the diagonal elements of the matrix. A matrix of eigenvectors R is also to be generated.

An identity matrix is used as a first approximation of R.

The initial off-diagonal norm is computed:

$$\nu_I = \left\{ \sum_{i \le k} 2A_{ik}^2 \right\}^{1/2} \qquad (1)$$

ν_I = initial norm

A = input matrix (symmetric)

This norm is divided by N at each stage to produce the threshold.

The final norm is computed:

$$\nu_F = \frac{\nu_I \times 10^{-6}}{N} \qquad (2)$$

This final norm is set sufficiently small that the requirement that any off-diagonal element A_{lm} shall be smaller than ν_F in absolute magnitude defines the convergence of the process.

An indicator is initialized. This indicator is later used to determine whether any off-diagonal elements have been found that are greater than the present threshold.

Each off-diagonal element is selected in turn and a transformation is performed to annihilate the off-diagonal (pivotal) element as shown by the following equations:

$$\lambda = -A_{lm} \qquad (3)$$

$$\mu = 1/2\,(A_{11} - A_{mm}) \qquad (4)$$

$$\omega = \text{sign}\,(\mu)\,\frac{\lambda}{\sqrt{\lambda^2 + \mu^2}} \qquad (5)$$

$$\sin\,\theta = \frac{\omega}{\sqrt{2\,(1 + \sqrt{1 - \omega^2})}} \qquad (6)$$

$$\cos\,\theta = \sqrt{1 - \sin^2\theta} \qquad (7)$$

$$B = A_{il} \cos\theta - A_{im} \sin\theta \quad (8)$$

$$C = A_{il} \sin\theta + A_{im} \cos\theta \quad (9)$$

$$B = R_{il} \cos\theta - R_{im} \sin\theta \quad (10)$$

$$R_{im} = R_{il} \sin\theta + R_{im} \cos\theta \quad (11)$$

$$R_{il} = B \quad (12)$$

$$A_{ll} = A_{il} \cos^2\theta + A_{mm} \sin^2\theta - 2A_{lm} \sin\theta \cos\theta \quad (13)$$

$$A_{mm} = A_{ll} \sin^2\theta + A_{mm} \cos^2\theta + 2A_{lm} \sin\theta \cos\theta \quad (14)$$

$$A_{lm} = (A_{ll} - A_{mm}) \sin\theta \cos\theta + A_{lm} (\cos^2\theta - \sin^2\theta) \quad (15)$$

The above calculations are repeated until all of the pivotal elements are less than the threshold.

Subroutine EIGEN

Purpose:

Compute eigenvalues and eigenvectors of a real symmetric matrix.

Usage:

CALL EIGEN(A, R, N, MV)

Description of parameters:

- 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 columnwise, 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:

Original matrix A must be real symmetric (storage mode=1). Matrix A cannot be in the same location as matrix R.

Subroutines and function subprograms required:

None.

Method:

Diagonalization method originated by Jacobi and adapted by von Neumann for large computers as found in 'Mathematical Methods for Digital Computers', edited by A. Ralston and H. S. Wilf, John Wiley and Sons, New York, 1962, Chapter 7.

```
      SUBROUTINE EIGEN(A,R,N,MV)                                        EIGEN  1
      DIMENSION A(1),R(1)                                               EIGEN  2
C        GENERATE IDENTITY MATRIX                                       EIGEN  3
      IF(MV-1) 10,25,10                                                 EIGEN  4
   10 IQ=-N                                                             EIGEN  5
      DO 20 J=1,N                                                       EIGEN  6
      IQ=IQ+N                                                           EIGEN  7
      DO 20 I=1,N                                                       EIGEN  8
      IJ=IQ+I                                                           EIGEN  9
      R(IJ)=0.0                                                         EIGEN 10
      IF(I-J) 20,15,20                                                  EIGEN 11
   15 R(IJ)=1.0                                                         EIGEN 12
   20 CONTINUE                                                          EIGEN 13
C        COMPUTE INITIAL AND FINAL NORMS (ANORM AND ANORMX)             EIGEN 14
   25 ANORM=0.0                                                         EIGEN 15
      DO 35 I=1,N                                                       EIGEN 16
      DO 35 J=I,N                                                       EIGEN 17
      IF(I-J) 30,35,30                                                  EIGEN 18
   30 IA=I+(J*J-J)/2                                                    EIGEN 19
      ANORM=ANORM+A(IA)*A(IA)                                           EIGEN 20
   35 CONTINUE                                                          EIGEN 21
      IF(ANORM) 165,165,40                                              EIGEN 22
   40 ANORM=1.414*SQRT(ANORM)                                           EIGEN 23
      ANRMX=ANORM*1.0E-6/FLOAT(N)                                       EIGEN 24
C        INITIALIZE INDICATORS AND COMPUTE THRESHOLD, THR               EIGEN 25
      IND=0                                                             EIGEN 26
      THR=ANORM                                                         EIGEN 27
   45 THR=THR/FLOAT(N)                                                  EIGEN 28
   50 L=1                                                               EIGEN 29
   55 M=L+1                                                             EIGEN 30
C        COMPUTE SIN AND COS                                            EIGEN 31
   60 MQ=(M*M-M)/2                                                      EIGEN 32
      LQ=(L*L-L)/2                                                      EIGEN 33
      LM=L+MQ                                                           EIGEN 34
   62 IF( ABS(A(LM))-THR) 130,65,65                                     EIGEN 35
   65 IND=1                                                             EIGEN 36
      LL=L+LQ                                                           EIGEN 37
      MM=M+MQ                                                           EIGEN 38
      X=0.5*(A(LL)-A(MM))                                               EIGEN 39
   68 Y=-A(LM)/ SQRT(A(LM)*A(LM)+X*X)                                   EIGEN 40
      IF(X) 70,75,75                                                    EIGEN 41
   70 Y=-Y                                                              EIGEN 42
   75 SINX=Y/ SQRT(2.0*(1.0+( SQRT(1.0-Y*Y))))                          EIGEN 43
      SINX2=SINX*SINX                                                   EIGEN 44
   78 COSX= SQRT(1.0-SINX2)                                             EIGEN 45
      COSX2=COSX*COSX                                                   EIGEN 46
      SINCS =SINX*COSX                                                  EIGEN 47
C        ROTATE L AND M COLUMNS                                         EIGEN 48
      ILQ=N*(L-1)                                                       EIGEN 49
      IMQ=N*(M-1)                                                       EIGEN 50
      DO 125 I=1,N                                                      EIGEN 51
      IQ=(I*I-I)/2                                                      EIGEN 52
      IF(I-L) 80,115,80                                                 EIGEN 53
   80 IF(I-M) 85,115,90                                                 EIGEN 54
   85 IM=I+MQ                                                           EIGEN 55
      GO TO 95                                                          EIGEN 56
   90 IM=M+IQ                                                           EIGEN 57
   95 IF(I-L) 100,105,105                                               EIGEN 58
  100 IL=I+LQ                                                           EIGEN 59
      GO TO 110                                                         EIGEN 60
  105 IL=L+IQ                                                           EIGEN 61
  110 X=A(IL)*COSX-A(IM)*SINX                                           EIGEN 62
      A(IM)=A(IL)*SINX+A(IM)*COSX                                       EIGEN 63
      A(IL)=X                                                           EIGEN 64
  115 IF(MV-1) 120,125,120                                              EIGEN 65
  120 ILR=ILQ+I                                                         EIGEN 66
      IMR=IMQ+I                                                         EIGEN 67
      X=R(ILR)*COSX-R(IMR)*SINX                                         EIGEN 68
      R(IMR)=R(ILR)*SINX+R(IMR)*COSX                                    EIGEN 69
      R(ILR)=X                                                          EIGEN 70
  125 CONTINUE                                                          EIGEN 71
      X=2.0*A(LM)*SINCS                                                 EIGEN 72
      Y=A(LL)*COSX2+A(MM)*SINX2-X                                       EIGEN 73
      X=A(LL)*SINX2+A(MM)*COSX2+X                                       EIGEN 74
      A(LM)=(A(LL)-A(MM))*SINCS+A(LM)*(COSX2-SINX2)                     EIGEN 75
      A(LL)=Y                                                           EIGEN 76
      A(MM)=X                                                           EIGEN 77
C        TESTS FOR COMPLETION                                           EIGEN 78
C        TEST FOR M = LAST COLUMN                                       EIGEN 79
  130 IF(M-N) 135,140,135                                               EIGEN 80
  135 M=M+1                                                             EIGEN 81
      GO TO 60                                                          EIGEN 82
C        TEST FOR L = SECOND FROM LAST COLUMN                           EIGEN 83
  140 IF(L-(N-1)) 145,150,145                                           EIGEN 84
  145 L=L+1                                                             EIGEN 85
      GO TO 55                                                          EIGEN 86
  150 IF(IND-1) 160,155,160                                             EIGEN 87
  155 IND=0                                                             EIGEN 88
      GO TO 50                                                          EIGEN 89
C        COMPARE THRESHOLD WITH FINAL NORM                              EIGEN 90
  160 IF(THR-ANRMX) 165,165,45                                          EIGEN 91
C        SORT EIGENVALUES AND EIGENVECTORS                              EIGEN 92
  165 IQ=-N                                                             EIGEN 93
      DO 185 I=1,N                                                      EIGEN 94
      IQ=IQ+N                                                           EIGEN 95
      LL=I+(I*I-I)/2                                                    EIGEN 96
      JQ=N*(I-2)                                                        EIGEN 97
      DO 185 J=I,N                                                      EIGEN 98
      JQ=JQ+N                                                           EIGEN 99
      MM=J+(J*J-J)/2                                                    EIGEN100
      IF(A(LL)-A(MM)) 170,185,185                                       EIGEN101
  170 X=A(LL)                                                           EIGEN102
      A(LL)=A(MM)                                                       EIGEN103
      A(MM)=X                                                           EIGEN104
      IF(MV-1) 175,185,175                                              EIGEN105
  175 DO 180 K=1,N                                                      EIGEN106
      ILR=IQ+K                                                          EIGEN107
      IMR=JQ+K                                                          EIGEN108
      X=R(ILR)                                                          EIGEN109
      R(ILR)=R(IMR)                                                     EIGEN110
  180 R(IMR)=X                                                          EIGEN111
  185 CONTINUE                                                          EIGEN112
      RETURN                                                            EIGEN113
      END                                                               EIGEN114
```

GMADD

Purpose:
Add two general matrices to form resultant general matrix.

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

Description of parameters:
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.

Subroutines and function subprograms required:
None.

Method:
Addition is performed element by element.

```
      SUBROUTINE GMADD(A,B,R,N,M)                                GMADD  1
      DIMENSION A(1),B(1),R(1)                                   GMADD  2
C        CALCULATE NUMBER OF ELEMENTS                            GMADD  3
      NM=N*M                                                     GMADD  4
C        ADD MATRICES                                            GMADD  5
      DO 10 I=1,NM                                               GMADD  6
   10 R(I)=A(I)+B(I)                                             GMADD  7
      RETURN                                                     GMADD  8
      END                                                        GMADD  9
```

GMSUB

Purpose:
Subtract one general matrix from another to form resultant matrix.

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

Description of parameters:
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.

Subroutines and function subprograms required:
None.

Method:
Matrix B elements are subtracted from corresponding matrix A elements.

```
      SUBROUTINE GMSUB(A,B,R,N,M)                                GMSUB  1
      DIMENSION A(1),B(1),R(1)                                   GMSUB  2
C        CALCULATE NUMBER OF ELEMENTS                            GMSUB  3
      NM=N*M                                                     GMSUB  4
C        SUBTRACT MATRICES                                       GMSUB  5
      DO 10 I=1,NM                                               GMSUB  6
   10 R(I)=A(I)-B(I)                                             GMSUB  7
      RETURN                                                     GMSUB  8
      END                                                        GMSUB  9
```

GMPRD

Purpose:
Multiply two general matrices to form a resultant general matrix.

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

Description of parameters:
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 matrix A.
Matrix R cannot be in the same location as matrix B.
Number of columns of matrix A must be equal to the number of rows of matrix B.

Subroutines and function subprograms required:
None.

Method:
The M by L matrix B is premultiplied by the N by M matrix A and the result is stored in the N by L matrix R.

```
      SUBROUTINE GMPRD(A,B,R,N,M,L)                                GMPRD  1
      DIMENSION A(1),B(1),R(1)                                     GMPRD  2
      IR=0                                                         GMPRD  3
      IK=-M                                                        GMPRD  4
      DO 10 K=1,L                                                  GMPRD  5
      IK=IK+M                                                      GMPRD  6
      DO 10 J=1,N                                                  GMPRD  7
      IR=IR+1                                                      GMPRD  8
      JI=J-N                                                       GMPRD  9
      IB=IK                                                        GMPRD 10
      R(IR)=0                                                      GMPRD 11
      DO 10 I=1,M                                                  GMPRD 12
      JI=JI+N                                                      GMPRD 13
      IB=IB+1                                                      GMPRD 14
   10 R(IR)=R(IR)+A(JI)*B(IB)                                      GMPRD 15
      RETURN                                                       GMPRD 16
      END                                                          GMPRD 17
```

GMTRA

Purpose:
Transpose a general matrix.

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

Description of parameters:
A - Name of matrix to be transposed.
R - Name of resultant matrix.
N - Number of rows of A and columns of R.
M - Number of columns of A and rows of R.

Remarks:
Matrix R cannot be in the same location as matrix A.
Matrices A and R must be stored as general matrices.

Subroutines and function subprograms required:
None.

Method:
Transpose N by M matrix A to form M by N matrix R.

```
      SUBROUTINE GMTRA(A,R,N,M)                                    GMTRA  1
      DIMENSION A(1),R(1)                                          GMTRA  2
      IR=0                                                         GMTRA  3
      DO 10 I=1,N                                                  GMTRA  4
      IJ=I-N                                                       GMTRA  5
      DO 10 J=1,M                                                  GMTRA  6
      IJ=IJ+N                                                      GMTRA  7
      IR=IR+1                                                      GMTRA  8
   10 R(IR)=A(IJ)                                                  GMTRA  9
      RETURN                                                       GMTRA 10
      END                                                          GMTRA 11
```

GTPRD

Purpose:

Premultiply a general matrix by the transpose of another general matrix.

Usage:

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

Description of parameters:

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:

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

Subroutines and function subprograms required:

None.

Method:

Matrix transpose of A is not actually calculated. Instead, elements of matrix A are taken columnwise rather than rowwise for postmultiplication by matrix B.

```
      SUBROUTINE GTPRD(A,B,R,N,M,L)                                     GTPRD  1
      DIMENSION A(1),B(1),R(1)                                          GTPRD  2
      IR=0                                                              GTPRD  3
      IK=-N                                                             GTPRD  4
      DO 10 K=1,L                                                       GTPRD  5
      IJ=0                                                              GTPRD  6
      IK=IK+N                                                           GTPRD  7
      DO 10 J=1,M                                                       GTPRD  8
      IB=IK                                                             GTPRD  9
      IR=IR+1                                                           GTPRD 10
      R(IR)=0                                                           GTPRD 11
      DO 10 I=1,N                                                       GTPRD 12
      IJ=IJ+1                                                           GTPRD 13
      IB=IB+1                                                           GTPRD 14
   10 R(IR)=R(IR)+A(IJ)*B(IB)                                           GTPRD 15
      RETURN                                                            GTPRD 16
      END                                                               GTPRD 17
```

MADD

Purpose:

Add two matrices element by element to form resultant matrix.

Usage:

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

Description of parameters:

A - Name of input matrix.
B - Name of 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 - Same as MSA except for matrix B.

Remarks:

None.

Subroutines and function subprograms required:

LOC

Method:

Storage mode of output matrix is first determined. Addition of corresponding elements is then performed.
The following table shows the storage mode of the output matrix for all combinations of input matrices:

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

```
      SUBROUTINE MADD(A,B,R,N,M,MSA,MSB)                                MADD   1
      DIMENSION A(1),B(1),R(1)                                          MADD   2
C        DETERMINE STORAGE MODE OF OUTPUT MATRIX                        MADD   3
      IF(MSA-MSB) 7,5,7                                                 MADD   4
    5 CALL LOC(N,M,NM,V,M,MSA)                                          MADD   5
      GO TO 100                                                         MADD   6
    7 MTEST=MSA*MSB                                                     MADD   7
      MSR=0                                                             MADD   8
      IF(MTEST) 20,20,10                                                MADD   9
   10 MSR=1                                                             MADD  10
   20 IF(MTEST-2) 35,35,30                                              MADD  11
   30 MSR=2                                                             MADD  12
C        LOCATE ELEMENTS AND PERFORM ADDITION                           MADD  13
   35 DO 90 J=1,M                                                       MADD  14
      DO 90 I=1,N                                                       MADD  15
      CALL LOC(I,J,IJR,N,M,MSR)                                         MADD  16
      IF(IJR) 40,90,40                                                  MADD  17
   40 CALL LOC(I,J,IJA,N,M,MSA)                                         MADD  18
      AEL=0.0                                                           MADD  19
      IF(IJA) 50,60,50                                                  MADD  20
   50 AEL=A(IJA)                                                        MADD  21
   60 CALL LOC(I,J,IJB,N,M,MSB)                                         MADD  22
      BEL=0.0                                                           MADD  23
      IF(IJB) 70,80,70                                                  MADD  24
   70 BEL=B(IJB)                                                        MADD  25
   80 R(IJR)=AEL+BEL                                                    MADD  26
   90 CONTINUE                                                          MADD  27
      RETURN                                                            MADD  28
C        ADD MATRICES FOR OTHER CASES                                   MADD  29
  100 DO 110 I=1,NM                                                     MADD  30
  110 R(I)=A(I)+B(I)                                                    MADD  31
      RETURN                                                            MADD  32
      END                                                               MADD  33
```

MSUB

Purpose:

Subtract two matrices element by element to form resultant matrix.

Usage:

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

Description of parameters:

- A - Name of input matrix.
- B - Name of 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 - Same as MSA except for matrix B.

Remarks:

None.

Subroutines and function subprograms required:

LOC

Method:

Structure of output matrix is first determined. Subtraction of matrix B elements from corresponding matrix A elements is then performed. The following table shows the storage mode of the output matrix for all combinations of input matrices:

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

```
      SUBROUTINE MSUB(A,B,R,N,M,MSA,MSB)                                MSUB   1
      DIMENSION A(1),B(1),R(1)                                          MSUB   2
C        DETERMINE STORAGE MODE OF OUTPUT MATRIX                        MSUB   3
      IF(MSA-MSB) 7,5,7                                                 MSUB   4
    5 CALL LOC(N,M,NM,N,M,MSA)                                          MSUB   5
      GO TO 100                                                         MSUB   6
    7 MTEST=MSA*MSB                                                     MSUB   7
      MSR=0                                                             MSUB   8
      IF(MTEST) 20,20,10                                                MSUB   9
   10 MSR=1                                                             MSUB  10
   20 IF(MTEST-2) 35,35,30                                              MSUB  11
   30 MSR=2                                                             MSUB  12
C        LOCATE ELEMENTS AND PERFORM SUBTRACTION                        MSUB  13
   35 DO 90 J=1,M                                                       MSUB  14
      DO 90 I=1,N                                                       MSUB  15
      CALL LOC(I,J,IJR,N,M,MSR)                                         MSUB  16
      IF(IJR) 40,90,40                                                  MSUB  17
   40 CALL LOC(I,J,IJA,N,M,MSA)                                         MSUB  18
      AEL=0.0                                                           MSUB  19
      IF(IJA) 50,60,50                                                  MSUB  20
   50 AEL=A(IJA)                                                        MSUB  21
   60 CALL LOC(I,J,IJB,N,M,MSB)                                         MSUB  22
      BEL=0.0                                                           MSUB  23
      IF(IJB) 70,80,70                                                  MSUB  24
   70 BEL=B(IJB)                                                        MSUB  25
   80 R(IJR)=AEL-BEL                                                    MSUB  26
   90 CONTINUE                                                          MSUB  27
      RETURN                                                            MSUB  28
C        SUBTRACT MATRICES FOR OTHER CASES                              MSUB  29
  100 DO 110 I=1,NM                                                     MSUB  30
  110 R(I)=A(I)-B(I)                                                    MSUB  31
      RETURN                                                            MSUB  32
      END                                                               MSUB  33
```

MPRD

Purpose:

Multiply two matrices to form a resultant matrix.

Usage:

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

Description of parameters:

- 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 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 and R.

Remarks:

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

Number of columns of matrix A must be equal to number of rows of matrix B.

Subroutines and function subprograms required:

LOC

Method:

The M by L matrix B is premultiplied by the N by M matrix A and the result is stored in the N by L matrix R. This is a row into column product.

The following table shows the storage mode of the output matrix for all combinations of input matrices:

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

```
      SUBROUTINE MPRD(A,B,R,N,M,MSA,MSB,L)                              MPRD   1
      DIMENSION A(1),B(1),R(1)                                          MPRD   2
C        SPECIAL CASE FOR DIAGONAL BY DIAGONAL                          MPRD   3
      MS=MSA*10+MSB                                                     MPRD   4
      IF(MS-22) 30,10,30                                                MPRD   5
   10 DO 20 I=1,N                                                       MPRD   6
   20 R(I)=A(I)*B(I)                                                    MPRD   7
      RETURN                                                            MPRD   8
C        ALL OTHER CASES                                                MPRD   9
   30 IR=1                                                              MPRD  10
      DO 90 K=1,L                                                       MPRD  11
      DO 90 J=1,N                                                       MPRD  12
      R(IR)=0                                                           MPRD  13
      DO 80 I=1,M                                                       MPRD  14
      IF(MS) 40,60,40                                                   MPRD  15
   40 CALL LOC(J,I,IA,N,M,MSA)                                          MPRD  16
      CALL LOC(I,K,IB,M,L,MSB)                                          MPRD  17
      IF(IA) 50,80,50                                                   MPRD  18
   50 IF(IB) 70,80,70                                                   MPRD  19
   60 IA=N*(I-1)+J                                                      MPRD  20
      IB=M*(K-1)+I                                                      MPRD  21
   70 R(IR)=R(IR)+A(IA)*B(IB)                                           MPRD  22
   80 CONTINUE                                                          MPRD  23
   90 IR=IR+1                                                           MPRD  24
      RETURN                                                            MPRD  25
      END                                                               MPRD  26
```

MTRA

Purpose:

Transpose a matrix.

Usage:

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

Description of parameters:

- A - Name of matrix to be transposed.
- R - Name of output matrix.
- N - Number of rows of A and columns of R.
- M - Number of columns of A and rows of 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:

Transpose 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, matrix R is the same as A.

```
      SUBROUTINE MTRA(A,R,N,M,MS)                                MTRA   1
      DIMENSION A(1),R(1)                                        MTRA   2
C        IF MS IS 1 OR 2, COPY A                                 MTRA   3
      IF(MS) 10,20,10                                            MTRA   4
   10 CALL MCPY(A,R,N,M,MS)                                      MTRA   5
      RETURN                                                     MTRA   6
C        TRANSPOSE GENERAL MATRIX                                MTRA   7
   20 IR=0                                                       MTRA   8
      DO 30 I=1,N                                                MTRA   9
      IJ=I-N                                                     MTRA  10
      DO 30 J=1,M                                                MTRA  11
      IJ=IJ+N                                                    MTRA  12
      IR=IR+1                                                    MTRA  13
   30 R(IR)=A(IJ)                                                MTRA  14
      RETURN                                                     MTRA  15
      END                                                        MTRA  16
```

TPRD

Purpose:

Transpose a matrix and postmultiply by another to form a resultant matrix.

Usage:

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

Description of parameters:

- 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 - Same as MSA except for matrix B.
- L - Number of columns in B and R.

Remarks:

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

Subroutines and function subprograms required:

LOC

Method:

Matrix transpose of A is not actually calculated. Instead, elements in matrix A are taken columnwise rather than rowwise for multiplication by matrix B.

The following table shows the storage mode of the output matrix for all combinations of input matrices:

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

```
      SUBROUTINE TPRD(A,B,R,N,M,MSA,MSB,L)                       TPRD   1
      DIMENSION A(1),B(1),R(1)                                   TPRD   2
C        SPECIAL CASE FOR DIAGONAL BY DIAGONAL                   TPRD   3
      MS=MSA*10+MSB                                              TPRD   4
      IF(MS-22) 30,10,30                                         TPRD   5
   10 DO 20 I=1,N                                                TPRD   6
   20 R(I)=A(I)*B(I)                                             TPRD   7
      RETURN                                                     TPRD   8
C        MULTIPLY TRANSPOSE OF A BY B                            TPRD   9
   30 IR=1                                                       TPRD  10
      DO 90 K=1,L                                                TPRD  11
      DO 90 J=1,M                                                TPRD  12
      R(IR)=0.0                                                  TPRD  13
      DO 80 I=1,N                                                TPRD  14
      IF(MS) 40,60,40                                            TPRD  15
   40 CALL LOC(I,J,IA,N,M,MSA)                                   TPRD  16
      CALL LOC(I,K,IB,N,L,MSB)                                   TPRD  17
      IF(IA) 50,80,50                                            TPRD  18
   50 IF(IB) 70,80,70                                            TPRD  19
   60 IA=N*(J-1)+I                                               TPRD  20
      IB=N*(K-1)+I                                               TPRD  21
   70 R(IR)=R(IR)+A(IA)*B(IB)                                    TPRD  22
   80 CONTINUE                                                   TPRD  23
   90 IR=IR+1                                                    TPRD  24
      RETURN                                                     TPRD  25
      END                                                        TPRD  26
```

MATA

Purpose:

Premultiply a matrix by its transpose to form a symmetric matrix.

Usage:

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

Description of parameters:

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

Remarks:

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

Matrix R is always a symmetric matrix with a storage mode=1.

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.

```
      SUBROUTINE MATA(A,R,N,M,MS)                                    MATA   1
      DIMENSION A(1),R(1)                                            MATA   2
      DO 60 K=1,M                                                    MATA   3
      KX=(K*K-K)/2                                                   MATA   4
      DO 60 J=1,M                                                    MATA   5
      IF(J-K) 10,10,60                                               MATA   6
   10 IR=J+KX                                                        MATA   7
      R(IR)=0                                                        MATA   8
      DO 60 I=1,N                                                    MATA   9
      IF(MS) 20,40,20                                                MATA  10
   20 CALL LOC(I,J,IA,N,M,MS)                                        MATA  11
      CALL LOC(I,K,IB,N,M,MS)                                        MATA  12
      IF(IA) 30,60,30                                                MATA  13
   30 IF(IB) 50,60,50                                                MATA  14
   40 IA=N*(J-1)+I                                                   MATA  15
      IB=N*(K-1)+I                                                   MATA  16
   50 R(IR)=R(IR)+A(IA)*A(IB)                                        MATA  17
   60 CONTINUE                                                       MATA  18
      RETURN                                                         MATA  19
      END                                                            MATA  20
```

SADD

Purpose:

Add a scalar to each element of a matrix to form a resultant matrix.

Usage:

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

Description of parameters:

- 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:

None.

Subroutines and function subprograms required:

LOC

Method:

Scalar is added to each element of matrix.

```
      SUBROUTINE SADD(A,C,R,N,M,MS)                                  SADD   1
      DIMENSION A(1),R(1)                                            SADD   2
C        COMPUTE VECTOR LENGTH, IT                                   SADD   3
      CALL LOC(N,M,IT,N,M,MS)                                        SADD   4
C        ADD SCALAR                                                  SADD   5
      DO 1 I=1,IT                                                    SADD   6
    1 R(I)=A(I)+C                                                    SADD   7
      RETURN                                                         SADD   8
      END                                                            SADD   9
```

SSUB

Purpose:

Subtract a scalar from each element of a matrix to form a resultant matrix.

Usage:

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

Description of parameters:

- 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:

None.

Subroutines and function subprograms required:

LOC

Method:

Scalar is subtracted from each element of matrix.

```
      SUBROUTINE SSUB(A,C,R,N,M,MS)                                SSUB   1
      DIMENSION A(1),R(1)                                          SSUB   2
C        COMPUTE VECTOR LENGTH, IT                                 SSUB   3
      CALL LOC(N,M,IT,N,M,MS)                                      SSUB   4
C        SUBTRACT SCALAR                                           SSUB   5
      DO 1 I=1,IT                                                  SSUB   6
    1 R(I)=A(I)-C                                                  SSUB   7
      RETURN                                                       SSUB   8
      END                                                          SSUB   9
```

SMPY

Purpose:

Multiply each element of a matrix by a scalar to form a resultant matrix.

Usage:

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

Description of parameters:

- 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:

None.

Subroutines and function subprograms required:

LOC

Method:

Scalar is multiplied by each element of matrix.

```
      SUBROUTINE SMPY(A,C,R,N,M,MS)                                SMPY   1
      DIMENSION A(1),R(1)                                          SMPY   2
C        COMPUTE VECTOR LENGTH, IT                                 SMPY   3
      CALL LOC(N,M,IT,N,M,MS)                                      SMPY   4
C        MULTIPLY BY SCALAR                                        SMPY   5
      DO 1 I=1,IT                                                  SMPY   6
    1 R(I)=A(I)*C                                                  SMPY   7
      RETURN                                                       SMPY   8
      END                                                          SMPY   9
```

SDIV

Purpose:

Divide each element of a matrix by a scalar to form a resultant matrix.

Usage:

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

Description of parameters:

- 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 scalar is zero, division is performed only once to cause floating-point overflow condition.

Subroutines and function subprograms required:

LOC

Method:

Each element of matrix is divided by scalar.

```
      SUBROUTINE SDIV(A,C,R,N,M,MS)                                    SDIV   1
      DIMENSION A(1),R(1)                                              SDIV   2
C        COMPUTE VECTOR LENGTH, IT                                     SDIV   3
      CALL LOC(N,M,IT,N,M,MS)                                          SDIV   4
C        DIVIDE BY SCALAR (IF SCALAR IS ZERO, DIVIDE ONLY ONCE)        SDIV   5
      IF(C) 2,1,2                                                      SDIV   6
    1 IT=1                                                             SDIV   7
    2 DO 3 I=1,IT                                                      SDIV   8
    3 R(I)=A(I)/C                                                      SDIV   9
      RETURN                                                           SDIV  10
      END                                                              SDIV  11
```

RADD

Purpose:

Add row of one matrix to row of another matrix.

Usage:

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

Description of parameters:

- 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:

Each element of row IRA of matrix A is added to corresponding element of row IRR of matrix R.

```
      SUBROUTINE RADD(A,IRA,R,IRR,N,M,MS,L)                            RADD   1
      DIMENSION A(1),R(1)                                              RADD   2
      IR=IRR-L                                                         RADD   3
      DO 2 J=1,M                                                       RADD   4
      IR=IR+L                                                          RADD   5
C        LOCATE INPUT ELEMENT FOR ANY MATRIX STORAGE MODE              RADD   6
      CALL LOC(IRA,J,IA,N,M,MS)                                        RADD   7
C        TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX                      RADD   8
      IF(IA) 1,2,1                                                     RADD   9
C        ADD ELEMENTS                                                  RADD  10
    1 R(IR)=R(IR)+A(IA)                                                RADD  11
    2 CONTINUE                                                         RADD  12
      RETURN                                                           RADD  13
      END                                                              RADD  14
```

CADD

Purpose:

Add column of one matrix to column of another matrix.

Usage:

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

Description of parameters:

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

Subroutines and function subprograms required:

LOC

Method:

Each element of column ICA of matrix A is added to corresponding element of column ICR of matrix R.

```
      SUBROUTINE CADD(A,ICA,R,ICR,N,M,MS,L)                          CADD   1
      DIMENSION A(1),R(1)                                            CADD   2
      IR=N*(ICR-1)                                                   CADD   3
      DO 2 I=1,N                                                     CADD   4
      IR=IR+1                                                        CADD   5
C        LOCATE INPUT ELEMENT FOR ANY MATRIX STORAGE MODE            CADD   6
      CALL LOC(I,ICA,IA,N,M,MS)                                      CADD   7
C        TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX                    CADD   8
      IF(IA) 1,2,1                                                   CADD   9
C        ADD ELEMENTS                                                CADD  10
    1 R(IR)=R(IR)+A(IA)                                              CADD  11
    2 CONTINUE                                                       CADD  12
      RETURN                                                         CADD  13
      END                                                            CADD  14
```

SRMA

Purpose:

Multiply row of matrix by a scalar and add to another row of the same matrix.

Usage:

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

Description of parameters:

- A - Name of matrix.
- C - Scalar.
- N - Number of rows in A.
- M - Number of columns in A.
- LA - Row in A to be multiplied by scalar.
- LB - Row in A to which product is added. If 0 is specified, product replaces elements in row LA.

Remarks:

Matrix A must be a general matrix.

Subroutines and function subprograms required:

None.

Method:

Each element of row LA is multiplied by scalar C and the product is added 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.

```
      SUBROUTINE SRMA(A,C,N,M,LA,LB)                                 SRMA   1
      DIMENSION A(1)                                                 SRMA   2
      LAJ=LA-N                                                       SRMA   3
      LBJ=LB-N                                                       SRMA   4
      DO 3 J=1,M                                                     SRMA   5
C        LOCATE ELEMENT IN BOTH ROWS                                 SRMA   6
      LAJ=LAJ+N                                                      SRMA   7
      LBJ=LBJ+N                                                      SRMA   8
C        CHECK LB FOR ZERO                                           SRMA   9
      IF(LB) 1,2,1                                                   SRMA  10
C        IF NOT, MULTIPLY BY CONSTANT AND ADD TO OTHER ROW           SRMA  11
    1 A(LBJ)=A(LAJ)*C+A(LBJ)                                         SRMA  12
      GO TO 3                                                        SRMA  13
C        OTHERWISE, MULTIPLY ROW BY CONSTANT                         SRMA  14
    2 A(LAJ)=A(LAJ)*C                                                SRMA  15
    3 CONTINUE                                                       SRMA  16
      RETURN                                                         SRMA  17
      END                                                            SRMA  18
```

SCMA

Purpose:
Multiply column of matrix by a scalar and add to another column of the same matrix.

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

Description of parameters:
- A - Name of matrix.
- C - Scalar.
- N - Number of rows in A.
- LA - Column in A to be multiplied by 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.

Subroutines and function subprograms required:
None.

Method:
Each element of column LA is multiplied by scalar C and the product is added to the corresponding element of column LB. 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.

```
      SUBROUTINE SCMA(A,C,N,LA,LB)                                      SCMA   1
      DIMENSION A(1)                                                    SCMA   2
C        LOCATE STARTING POINT OF BOTH COLUMNS                          SCMA   3
      LA=N*(LA-1)                                                       SCMA   4
      LB=N*(LB-1)                                                       SCMA   5
      DO 3 I=1,N                                                        SCMA   6
      LA=LA+1                                                           SCMA   7
      LB=LB+1                                                           SCMA   8
C        CHECK LB FOR ZERO                                              SCMA   9
      IF(LB) 1,2,1                                                      SCMA  10
C        IF NOT MULTIPLY BY CONSTANT AND ADD TO SECOND COLUMN           SCMA  11
    1 A(LB)=A(LA)*C+A(LB)                                               SCMA  12
      GO TO 3                                                           SCMA  13
C        OTHERWISE, MULTIPLY COLUMN BY CONSTANT                         SCMA  14
    2 A(LA)=A(LA)*C                                                     SCMA  15
    3 CONTINUE                                                          SCMA  16
      RETURN                                                            SCMA  17
      END                                                               SCMA  18
```

RINT

Purpose:
Interchange two rows of a matrix.

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

Description of parameters:
- 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.

Subroutines and function subprograms required:
None.

Method:
Each element of row LA is interchanged with corresponding element of row LB.

```
      SUBROUTINE RINT(A,N,M,LA,LB)                                      RINT   1
      DIMENSION A(1)                                                    RINT   2
      LAJ=LA-N                                                          RINT   3
      LBJ=LB-N                                                          RINT   4
      DO 3 J=1,M                                                        RINT   5
C        LOCATE ELEMENTS IN BOTH ROWS                                   RINT   6
      LAJ=LAJ+N                                                         RINT   7
      LBJ=LBJ+N                                                         RINT   8
C        INTERCHANGE ELEMENTS                                           RINT   9
      SAVE=A(LAJ)                                                       RINT  10
      A(LAJ)=A(LBJ)                                                     RINT  11
    3 A(LBJ)=SAVE                                                       RINT  12
      RETURN                                                            RINT  13
      END                                                               RINT  14
```

CINT

Purpose:
Interchange two columns of a matrix.

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

Description of parameters:
- 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.

Subroutines and function subprograms required:
None.

Method:
Each element of column LA is interchanged with corresponding element of column LB.

```
      SUBROUTINE CINT(A,N,LA,LB)                                        CINT   1
      DIMENSION A(1)                                                    CINT   2
C       LOCATE STARTING POINT OF BOTH COLUMNS                           CINT   3
      LA=N*(LA-1)                                                       CINT   4
      LB=N*(LB-1)                                                       CINT   5
      DO 3 I=1,N                                                        CINT   6
      LA=LA+1                                                           CINT   7
      LB=LB+1                                                           CINT   8
C       INTERCHANGE ELEMENTS                                            CINT   9
      SAVE=A(LA)                                                        CINT  10
      A(LA)=A(LB)                                                       CINT  11
    3 A(LB)=SAVE                                                        CINT  12
      RETURN                                                            CINT  13
      END                                                               CINT  14
```

RSUM

Purpose:
Sum elements of each row to form column vector.

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

Description of parameters:
- 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:
Elements are summed across each row into a corresponding element of output column vector R.

```
      SUBROUTINE RSUM(A,R,N,M,MS)                                       RSUM   1
      DIMENSION A(1),R(1)                                               RSUM   2
      DO 3 I=1,N                                                        RSUM   3
C       CLEAR OUTPUT LOCATION                                           RSUM   4
      R(I)=0.0                                                          RSUM   5
      DO 3 J=1,M                                                        RSUM   6
C       LOCATE ELEMENT FOR ANY MATRIX STORAGE MODE                      RSUM   7
      CALL LOC(I,J,IJ,N,M,MS)                                           RSUM   8
C       TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX                        RSUM   9
      IF(IJ) 2,3,2                                                      RSUM  10
C       ACCUMULATE IN OUTPUT VECTOR                                     RSUM  11
    2 R(I)=R(I)+A(IJ)                                                   RSUM  12
    3 CONTINUE                                                          RSUM  13
      RETURN                                                            RSUM  14
      END                                                               RSUM  15
```

CSUM

Purpose:

Sum elements of each column to form row vector.

Usage:

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

Description of parameters:

- A - Name of input matrix.
- R - Name of 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.

Remarks:

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

Subroutines and function subprograms required:

LOC

Method:

Elements are summed down each column into a corresponding element of output row vector R.

```
      SUBROUTINE CSUM(A,R,N,M,MS)                                       CSUM   1
      DIMENSION A(1),R(1)                                               CSUM   2
      DO 3 J=1,M                                                        CSUM   3
C        CLEAR OUTPUT LOCATION                                          CSUM   4
      R(J)=0.0                                                          CSUM   5
      DO 3 I=1,N                                                        CSUM   6
C        LOCATE ELEMENT FOR ANY MATRIX STORAGE MODE                     CSUM   7
      CALL LOC(I,J,IJ,N,M,MS)                                           CSUM   8
C        TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX                       CSUM   9
      IF(IJ) 2,3,2                                                      CSUM  10
C        ACCUMULATE IN OUTPUT VECTOR                                    CSUM  11
    2 R(J)=R(J)+A(IJ)                                                   CSUM  12
    3 CONTINUE                                                          CSUM  13
      RETURN                                                            CSUM  14
      END                                                               CSUM  15
```

RTAB

The function of this subroutine is graphically displayed by Figure 6 (see description under "Method").

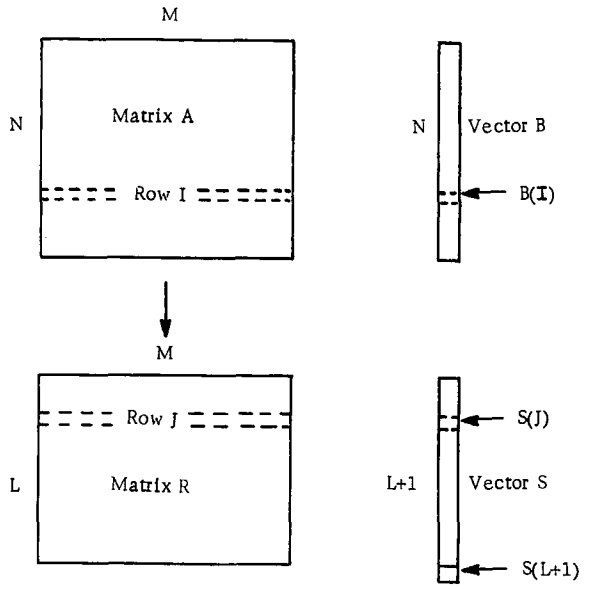


Figure 6. Row tabulation

Subroutine RTAB

Purpose:

Tabulate rows of a matrix to form a summary matrix.

Usage:

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

Description of parameters:

- 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 this subroutine.
- S - Name of output vector of 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 is always a general matrix.

Subroutines and function subprograms required:

LOC
RADD

Method:

Rows of data in matrix A are tabulated using the key contained in vector B. The floating point number in B(I) is truncated to form J. The I^{th} row of A is added to the J^{th} row of R, element by element, and one is added to S(J). If J is not between one and L, one 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 used to form the corresponding row of R. Element S(L+1) contains a count of the number of rows of A not included in R as a result of J being less than one or greater than L.

```
      SUBROUTINE RTAB(A,B,R,S,N,M,MS,L)                                 RTAB   1
      DIMENSION A(1),B(1),R(1),S(1)                                     RTAB   2
C        CLEAR OUTPUT AREAS                                             RTAB   3
      CALL LOC(M,L,IT,4,L,0)                                            RTAB   4
      DO 10 IR=1,IT                                                     RTAB   5
   10 R(IR)=0.0                                                         RTAB   6
      DO 20 IS=1,L                                                      RTAB   7
   20 S(IS)=0.0                                                         RTAB   8
      S(L+1)=0.0                                                        RTAB   9
      DO 60 I=1,N                                                       RTAB  10
C        TEST FOR THE KEY OUTSIDE THE RANGE                             RTAB  11
      IF(B(I)) 50,50,30                                                 RTAB  12
   30 E=L                                                               RTAB  13
      IF(B(I)-E) 40,40,50                                               RTAB  14
   40 JR=B(I)                                                           RTAB  15
C        ADD ROW OF A TO ROW OF R AND 1 TO COUNT                        RTAB  16
      CALL RADD(A,I,R,JR,N,M,MS,L)                                      RTAB  17
      S(JR)=S(JR)+1.0                                                   RTAB  18
      GO TO 60                                                          RTAB  19
   50 S(L+1)=S(L+1)+1.0                                                 RTAB  20
   60 CONTINUE                                                          RTAB  21
      RETURN                                                            RTAB  22
      END                                                               RTAB  23
```

CTAB

The function of this subroutine is graphically displayed by Figure 7 (see description under "Method").

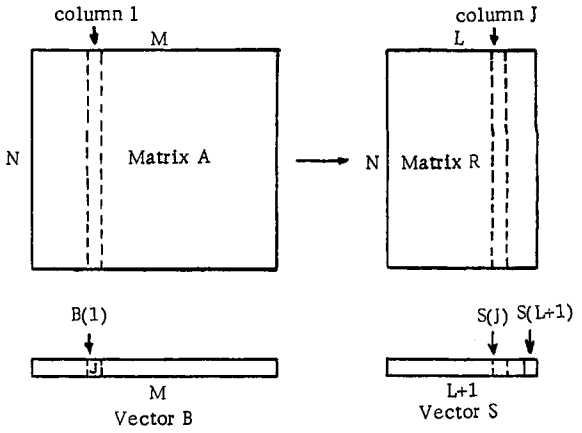


Figure 7. Column tabulation

Subroutine CTAB

Purpose:

Tabulate columns of a matrix to form a summary matrix.

Usage:

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

Description of parameters:

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 for storage mode of matrix A:
  0 - General.
  1 - Symmetric.
  2 - Diagonal.

Remarks:

Matrix R is always a general matrix.

Subroutines and function subprograms required:

LOC
CADD

Method:

Columns of data in matrix A are tabulated using the key contained in vector B. The floating-point number in B(I) is truncated to form J. The I^{th} column of A is added to the J^{th} column of matrix R and one is added to S(J). If the value of J is not between one and M, one 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 one or greater than L.

```
      SUBROUTINE CTAB(A,B,R,S,N,M,MS,L)                                 CTAB   1
      DIMENSION A(1),B(1),R(1),S(1)                                     CTAB   2
C        CLEAR OUTPUT AREAS                                             CTAB   3
      CALL LOC(N,L,IT,N,L,0)                                            CTAB   4
      DO 10 IR=1,IT                                                     CTAB   5
   10 R(IR)=0.0                                                         CTAB   6
      DO 20 IS=1,L                                                      CTAB   7
   20 S(IS)=0.0                                                         CTAB   8
      S(L+1)=0.0                                                        CTAB   9
      DO 60 I=1,M                                                       CTAB  10
C        TEST FOR THE KEY OUTSIDE THE RANGE                             CTAB  11
      IF(B(I)) 50,50,30                                                 CTAB  12
   30 E=L                                                               CTAB  13
      IF(B(I)-E) 40,40,50                                               CTAB  14
   40 JR=B(I)                                                           CTAB  15
C        ADD COLUMN OF A TO COLUMN OF R AND 1 TO COUNT                  CTAB  16
      CALL CADD(A,I,R,JR,N,M,MS,L)                                      CTAB  17
      S(JR)=S(JR)+1.0                                                   CTAB  18
      GO TO 60                                                          CTAB  19
   50 S(L+1)=S(L+1)+1.0                                                 CTAB  20
   60 CONTINUE                                                          CTAB  21
      RETURN                                                            CTAB  22
      END                                                               CTAB  23
```

RSRT

Purpose:

Sort rows of a matrix.

Usage:

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

Description of parameters:

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

Remarks:

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

Matrix R is always a general matrix..

N must be greater than 1. This routine sorts into ascending order. Sorting into descending order requires changing card RSRT 013 to read IF(R(I-1)-R(I)) 30, 40, 40

Subroutines and function subprograms required:

LOC

Method:

Rows of input matrix A are sorted to form output matrix R. The sorted row sequence is determined by the values of elements in column vector B. The lowest valued element in B will cause the corresponding row of A to be placed in the first row of R. The highest valued element of B will cause the corresponding row of A to be placed in the last row of R. If duplicate values exist in B, the corresponding rows of A are moved to R in the same order as in A.

```
      SUBROUTINE RSRT(A,B,R,N,M,MS)                                     RSRT   1
      DIMENSION A(1),B(1),R(1)                                          RSRT   2
C        MOVE SORTING KEY VECTOR TO FIRST COLUMN OF OUTPUT MATRIX       RSRT   3
C        AND BUILD ORIGINAL SEQUENCE LIST IN SECOND COLUMN              RSRT   4
      DO 10 I=1,N                                                       RSRT   5
      R(I)=B(I)                                                         RSRT   6
      I2=I+N                                                            RSRT   7
   10 R(I2)=I                                                           RSRT   8
C        SORT ELEMENTS IN SORTING KEY VECTOR (ORIGINAL SEQUENCE LIST    RSRT   9
C        IS RESEQUENCED ACCORDINGLY)                                    RSRT  10
      L=N+1                                                             RSRT M01
   20 ISORT=0                                                           RSRT  11
      L=L-1                                                             RSRT M02
      DO 40 I=2,L                                                       RSRT M03
      IF(R(I)-R(I-1)) 30,40,40                                          RSRT  13
   30 ISORT=1                                                           RSRT M04
      RSAVE=R(I)                                                        RSRT  15
      R(I)=R(I-1)                                                       RSRT  16
      R(I-1)=RSAVE                                                      RSRT  17
      I2=I+N                                                            RSRT  18
      SAVER=R(I2)                                                       RSRT  19
      R(I2)=R(I2-1)                                                     RSRT  20
      R(I2-1)=SAVER                                                     RSRT  21
   40 CONTINUE                                                          RSRT  22
      IF(ISORT) 20,50,20                                                RSRT  23
C        MOVE ROWS FROM MATRIX A TO MATRIX R (NUMBER IN SECOND COLUMN   RSRT  24
C        OF R REPRESENTS ROW NUMBER OF MATRIX A TO BE MOVED)            RSRT  25
   50 DO 80 I=1,N                                                       RSRT  26
C        GET ROW NUMBER IN MATRIX A                                     RSRT  27
      I2=I+N                                                            RSRT  28
      IN=R(I2)                                                          RSRT  29
      IR=I-N                                                            RSRT  30
      DO 80 J=1,M                                                       RSRT  31
C        LOCATE ELEMENT IN OUTPUT MATRIX                                RSRT  32
      IR=IR+N                                                           RSRT  33
C        LOCATE ELEMENT IN INPUT MATRIX                                 RSRT  34
      CALL LOC(IN,J,IA,N,M,MS)                                          RSRT  35
C        TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX                       RSRT  36
      IF(IA) 60,70,60                                                   RSRT  37
C        MOVE ELEMENT TO OUTPUT MATRIX                                  RSRT  38
   60 R(IR)=A(IA)                                                       RSRT  39
      GO TO 80                                                          RSRT  40
   70 R(IR)=0                                                           RSRT  41
   80 CONTINUE                                                          RSRT  42
      RETURN                                                            RSRT  43
      END                                                               RSRT  44
```

CSRT

Purpose:

Sort columns of a matrix.

Usage:

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

Description of parameters:

- 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 cannot be in the same location as matrix A.

Matrix R is always a general matrix.

N must be greater than 1. This routine sorts into ascending order. Sorting into descending order requires changing card CSRT 016 to read IF(R(IP)-R(IQ)) 30, 40, 40

Subroutines and function subprograms required:

LOC

CCPY

Method:

Columns of input matrix A are sorted to form output matrix R. The sorted column sequence is determined by the values of elements in row vector B. The lowest valued element in B will cause the corresponding column of A to be placed in the first column of R. The highest valued element of B will cause the corresponding row of A to be placed in the last column of R. If duplicate values exist in B, the corresponding columns of A are moved to R in the same order as in A.

```
      SUBROUTINE CSRT(A,B,R,N,M,MS)                                      CSRT   1
      DIMENSION A(1),B(1),R(1)                                           CSRT   2
C        MOVE SORTING KEY VECTOR TO FIRST ROW OF OUTPUT MATRIX           CSRT   3
C        AND BUILD ORIGINAL SEQUENCE LIST IN SECOND ROW                  CSRT   4
      IK=1                                                               CSRT   5
      DO 10 J=1,M                                                        CSRT   6
      R(IK)=B(J)                                                         CSRT   7
      R(IK+1)=J                                                          CSRT   8
   10 IK=IK+N                                                            CSRT   9
C        SORT ELEMENTS IN SORTING KEY VECTOR (ORIGINAL SEQUENCE LIST     CSRT  10
C        IS RESEQUENCED ACCORDINGLY)                                     CSRT  11
      L=M+1                                                              CSRT M01
   20 ISORT=0                                                            CSRT  12
      L=L-1                                                              CSRT M02
      IP=1                                                               CSRT  13
      IQ=N+1                                                             CSRT  14
      DO 50 J=2,L                                                        CSRT M03
      IF(R(IQ)-R(IP)) 30,40,40                                           CSRT  16
   30 ISORT=1                                                            CSRT M04
      RSAVE=R(IQ)                                                        CSRT  18
      R(IQ)=R(IP)                                                        CSRT  19
      R(IP)=RSAVE                                                        CSRT  20
      SAVER=R(IQ+1)                                                      CSRT  21
      R(IQ+1)=R(IP+1)                                                    CSRT  22
      R(IP+1)=SAVER                                                      CSRT  23
   40 IP=IP+N                                                            CSRT  24
      IQ=IQ+N                                                            CSRT  25
   50 CONTINUE                                                           CSRT  26
      IF(ISORT) 20,60,20                                                 CSRT  27
C        MOVE COLUMNS FROM MATRIX A TO MATRIX R (NUMBER IN SECOND ROW    CSRT  28
C        OF R REPRESENTS COLUMN NUMBER OF MATRIX A TO BE MOVED)          CSRT  29
   60 IQ=-N                                                              CSRT  30
      DO 70 J=1,M                                                        CSRT  31
      IQ=IQ+N                                                            CSRT  32
C        GET COLUMN NUMBER IN MATRIX A                                   CSRT  33
      I2=IQ+2                                                            CSRT  34
      IN=R(I2)                                                           CSRT  35
C        MOVE COLUMN                                                     CSRT  36
      IR=IQ+1                                                            CSRT  37
      CALL CCPY(A,IN,R(IR),N,M,MS)                                       CSRT  38
   70 CONTINUE                                                           CSRT  39
      RETURN                                                             CSRT  40
      END                                                                CSRT  41
```

RCUT

Purpose:

Partition a matrix between specified rows to form two resultant matrices.

Usage:

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

Description of parameters:

- 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 cannot be in same location as matrix A.
Matrix S cannot be in same location as matrix A.
Matrix R cannot be in same location as matrix S.
Matrix R and matrix S are always general matrices.

Subroutines and function subprograms required:

LOC

Method:

Elements of matrix A above row L are moved to form matrix R of L-1 rows and M columns. Elements of matrix A in row L and below are moved to form matrix S of N-L+1 rows and M columns.

```
      SUBROUTINE RCUT(A,L,R,S,N,M,MS)                                   RCUT   1
      DIMENSION A(1),R(1),S(1)                                          RCUT   2
      IR=0                                                              RCUT   3
      IS=0                                                              RCUT   4
      DO 70 J=1,M                                                       RCUT   5
      DO 70 I=1,N                                                       RCUT   6
C     .    FIND LOCATION IN OUTPUT MATRIX AND SET TO ZERO               RCUT   7
      IF(I-L) 20,10,10                                                  RCUT   8
   10 IS=IS+1                                                           RCUT   9
      S(IS)=0.0                                                         RCUT  10
      GO TO 30                                                          RCUT  11
   20 IR=IR+1                                                           RCUT  12
      R(IR)=0.0                                                         RCUT  13
C        LOCATE ELEMENT FOR ANY MATRIX STORAGE MODE                     RCUT  14
   30 CALL LOC(I,J,IJ,N,M,MS)                                           RCUT  15
C        TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX                       RCUT  16
      IF(IJ) 40,70,40                                                   RCUT  17
C        DETERMINE WHETHER ABOVE OR BELOW L                             RCUT  18
   40 IF(I-L) 60,50,50                                                  RCUT  19
   50 S(IS)=A(IJ)                                                       RCUT  20
      GO TO 70                                                          RCUT  21
   60 R(IR)=A(IJ)                                                       RCUT  22
   70 CONTINUE                                                          RCUT  23
      RETURN                                                            RCUT  24
      END                                                               RCUT  25
```

CCUT

Purpose:

Partition a matrix between specified columns to form two resultant matrices.

Usage:

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

Description of parameters:

- 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.
- 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 cannot be in same location as matrix A.
Matrix S cannot be in same location as matrix A.
Matrix R cannot be in same location as matrix S.
Matrix R and matrix S are always general matrices.

Subroutines and function subprograms required:

LOC

Method:

Elements of matrix A to the left of column L are moved to form matrix R of N rows and L-1 columns. Elements of matrix A in column L and to the right of L are moved to form matrix S of N rows and M-L+1 columns.

```
      SUBROUTINE CCUT(A,L,R,S,N,M,MS)                                   CCUT   1
      DIMENSION A(1),R(1),S(1)                                          CCUT   2
      IR=0                                                              CCUT   3
      IS=0                                                              CCUT   4
      DO 70 J=1,M                                                       CCUT   5
      DO 70 I=1,N                                                       CCUT   6
C        FIND LOCATION IN OUTPUT MATRIX AND SET TO ZERO                 CCUT   7
      IF(J-L) 20,10,10                                                  CCUT   8
   10 IS=IS+1                                                           CCUT   9
      S(IS)=0.0                                                         CCUT  10
      GO TO 30                                                          CCUT  11
   20 IR=IR+1                                                           CCUT  12
      R(IR)=0.0                                                         CCUT  13
C        LOCATE ELEMENT FOR ANY MATRIX STORAGE MODE                     CCUT  14
   30 CALL LOC(I,J,IJ,N,M,MS)                                           CCUT  15
C        TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX                       CCUT  16
      IF(IJ) 40,70,40                                                   CCUT  17
C        DETERMINE WHETHER RIGHT OR LEFT OF L                           CCUT  18
   40 IF(J-L) 60,50,50                                                  CCUT  19
   50 S(IS)=A(IJ)                                                       CCUT  20
      GO TO 70                                                          CCUT  21
   60 R(IR)=A(IJ)                                                       CCUT  22
   70 CONTINUE                                                          CCUT  23
      RETURN                                                            CCUT  24
      END                                                               CCUT  25
```

RTIE

Purpose:

Adjoin two matrices with same column dimension to form one resultant matrix. (See Method.)

Usage:

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

Description of parameters:

- 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 - Same as MSA except for matrix B.
- L - Number of rows in B.

Remarks:

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

Subroutines and function subprograms required:

LOC

Method:

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

```
      SUBROUTINE RTIE(A,B,R,N,M,MSA,MSB,L)                              RTIE   1
      DIMENSION A(1),B(1),R(1)                                          RTIE   2
      NN=N                                                              RTIE   3
      IR=0                                                              RTIE   4
      NX=NN                                                             RTIE   5
      MSX=MSA                                                           RTIE   6
      DO 9 J=1,M                                                        RTIE   7
      DO 8 II=1,2                                                       RTIE   8
      DO 7 I=1,NN                                                       RTIE   9
      IR=IR+1                                                           RTIE  10
      R(IR)=0.0                                                         RTIE  11
C        LOCATE ELEMENT FOR ANY MATRIX STORAGE MODE                     RTIE  12
      CALL LOC(I,J,IJ,NN,M,MSX)                                         RTIE  13
C        TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX                       RTIE  14
      IF(IJ) 2,7,2                                                      RTIE  15
C        MOVE ELEMENT TO MATRIX R                                       RTIE  16
    2 GO TO(3,4),II                                                     RTIE  17
    3 R(IR)=A(IJ)                                                       RTIE  18
      GO TO 7                                                           RTIE  19
    4 R(IR)=B(IJ)                                                       RTIE  20
    7 CONTINUE                                                          RTIE  21
C        REPEAT ABOVE FOR MATRIX B                                      RTIE  22
      MSX=MSB                                                           RTIE  23
    8 NN=L                                                              RTIE  24
C        RESET FOR NEXT COLUMN                                          RTIE  25
      MSX=MSA                                                           RTIE  26
    9 NN=NX                                                             RTIE  27
      RETURN                                                            RTIE  28
      END                                                               RTIE  29
```

CTIE

Purpose:

Adjoin two matrices with same row dimension to form one resultant matrix. (See Method.)

Usage:

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

Description of parameters:

- 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 cannot be in the same location as matrices A or B.
Matrix R is always a general matrix.
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.

```
      SUBROUTINE CTIE(A,B,R,N,M,MSA,MSB,L)                              CTIE   1
      DIMENSION A(1),B(1),R(1)                                          CTIF   2
      MM=M                                                              CTIE   3
      IR=0                                                              CTIE   4
      MSX=MSA                                                           CTIE   5
      DO 6 JJ=1,2                                                       CTIE   6
      DO 5 J=1,MM                                                       CTIE   7
      DO 5 I=1,N                                                        CTIE   8
      IR=IR+1                                                           CTIE   9
      R(IR)=0.0                                                         CTIE  10
C        LOCATE ELEMENT FOR ANY MATRIX STORAGE MODE                     CTIE  11
      CALL LOC(I,J,IJ,N,MM,MSX)                                         CTIF  12
C        TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX                       CTIE  13
      IF(IJ) 2,5,2                                                      CTIE  14
C        MOVE ELEMENT TO MATRIX R                                       CTIE  15
    2 GO TO(3,4),JJ                                                     CTIE  16
    3 R(IR)=A(IJ)                                                       CTIE  17
      GO TO 5                                                           CTIF  18
    4 R(IR)=B(IJ)                                                       CTIF  19
    5 CONTINUE                                                          CTIE  20
C        REPEAT ABOVE FOR MATRIX B                                      CTIE  21
      MSX=MSB                                                           CTIF  22
      MM=L                                                              CTIE  23
    6 CONTINUE                                                          CTIE  24
      RETURN                                                            CTIE  25
      END                                                               CTIE  26
```

MCPY

Purpose:
Copy entire matrix.

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

Description of parameters:
- 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.

Remarks:
None.

Subroutines and function subprograms required:
LOC

Method:
Each element of matrix A is moved to the corresponding element of matrix R.

```
      SUBROUTINE MCPY(A,R,N,M,MS)                                    MCPY   1
      DIMENSION A(1),R(1)                                            MCPY   2
C        COMPUTE VECTOR LENGTH, IT                                   MCPY   3
      CALL LOC(N,M,IT,N,M,MS)                                        MCPY   4
C        COPY MATRIX                                                 MCPY   5
      DO 1 I=1,IT                                                    MCPY   6
    1 R(I)=A(I)                                                      MCPY   7
      RETURN                                                         MCPY   8
      END                                                            MCPY   9
```

XCPY

Purpose:
Copy a portion of a matrix.

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

Description of parameters:
- 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 cannot be in the same location as matrix A.
Matrix R is always a general matrix.

Subroutines and function subprograms required:
LOC

Method:
Matrix R is formed 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.

```
      SUBROUTINE XCPY(A,R,L,K,NR,MR,NA,MA,MS)                        XCPY   1
      DIMENSION A(1),R(1)                                            XCPY   2
C        INITIALIZE                                                  XCPY   3
      IR=0                                                           XCPY   4
      L2=L+NR-1                                                      XCPY   5
      K2=K+MR-1                                                      XCPY   6
      DO 5 J=K,K2                                                    XCPY   7
      DO 5 I=L,L2                                                    XCPY   8
      IR=IR+1                                                        XCPY   9
      R(IR)=0.0                                                      XCPY  10
C        LOCATE ELEMENT FOR ANY MATRIX STORAGE MODE                  XCPY  11
      CALL LOC(I,J,IA,NA,MA,MS)                                      XCPY  12
C        TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX                    XCPY  13
      IF(IA) 4,5,4                                                   XCPY  14
    4 R(IR)=A(IA)                                                    XCPY  15
    5 CONTINUE                                                       XCPY  16
      RETURN                                                         XCPY  17
      END                                                            XCPY  18
```

RCPY

Purpose:
Copy specified row of a matrix into a vector.

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

Description of parameters:
- 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.

Remarks:
None.

Subroutines and function subprograms required:
LOC

Method:
Elements of row L are moved to corresponding positions of vector R.

```
      SUBROUTINE RCPY(A,L,R,N,M,MS)                                    RCPY   1
      DIMENSION A(1),R(1)                                              RCPY   2
      DO 3 J=1,M                                                       RCPY   3
C        LOCATE ELEMENT FOR ANY MATRIX STORAGE MODE                    RCPY   4
      CALL LOC(L,J,LJ,N,M,MS)                                          RCPY   5
C        TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX                      RCPY   6
      IF(LJ) 1,2,1                                                     RCPY   7
C        MOVE ELEMENT TO R                                             RCPY   8
    1 R(J)=A(LJ)                                                       RCPY   9
      GO TO 3                                                          RCPY  10
    2 R(J)=0.0                                                         RCPY  11
    3 CONTINUE                                                         RCPY  12
      RETURN                                                           RCPY  13
      END                                                              RCPY  14
```

CCPY

Purpose:
Copy specified column of a matrix into a vector.

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

Description of parameters:
- 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.

Remarks:
None.

Subroutines and function subprograms required:
LOC

Method:
Elements of column L are moved to corresponding positions of vector R.

```
      SUBROUTINE CCPY(A,L,R,N,M,MS)                                    CCPY   1
      DIMENSION A(1),R(1)                                              CCPY   2
      DO 3 I=1,N                                                       CCPY   3
C        LOCATE ELEMENT FOR ANY MATRIX STORAGE MODE                    CCPY   4
      CALL LOC(I,L,IL,N,M,MS)                                          CCPY   5
C        TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX                      CCPY   6
      IF(IL) 1,2,1                                                     CCPY   7
C        MOVE ELEMENT TO R                                             CCPY   8
    1 R(I)=A(IL)                                                       CCPY   9
      GO TO 3                                                          CCPY  10
    2 R(I)=0.0                                                         CCPY  11
    3 CONTINUE                                                         CCPY  12
      RETURN                                                           CCPY  13
      END                                                              CCPY  14
```

DCPY

Purpose:
Copy diagonal elements of a matrix into a vector.

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

Description of parameters:
- 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:
Input matrix must be a square matrix.

Subroutines and function subprograms required:
LOC

Method:
Elements on diagonal of matrix are moved to corresponding positions of vector R.

```
      SUBROUTINE DCPY(A,R,N,MS)                                      DCPY   1
      DIMENSION A(1),R(1)                                            DCPY   2
      DO 3 J=1,N                                                     DCPY   3
C        LOCATE DIAGONAL ELEMENT FOR ANY MATRIX STORAGE MODE         DCPY   4
      CALL LOC(J,J,IJ,N,N,MS)                                        DCPY   5
C        MOVE DIAGONAL ELEMENT TO VECTOR R                           DCPY   6
    3 R(J)=A(IJ)                                                     DCPY   7
      RETURN                                                         DCPY   8
      END                                                            DCPY   9
```

SCLA

Purpose:
Set each element of a matrix equal to a given scalar.

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

Description of parameters:
- 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.

Remarks:
None.

Subroutines and function subprograms required:
LOC

Method:
Each element of matrix A is replaced by Scalar C.

```
      SUBROUTINE SCLA(A,C,N,M,MS)                                    SCLA   1
      DIMENSION A(1)                                                 SCLA   2
C        COMPUTE VECTOR LENGTH, IT                                   SCLA   3
      CALL LOC(N,M,IT,N,M,MS)                                        SCLA   4
C        REPLACE BY SCALAR                                           SCLA   5
      DO 1 I=1,IT                                                    SCLA   6
    1 A(I)=C                                                         SCLA   7
      RETURN                                                         SCLA   8
      END                                                            SCLA   9
```

DCLA

Purpose:

Set each diagonal element of a matrix equal to a scalar.

Usage:

CALL DCLA (A, C, N, MS)

Description of parameters:

- A - Name of input matrix.
- C - Scalar.
- 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:

Each element on diagonal of matrix is replaced by scalar C.

```
      SUBROUTINE DCLA(A,C,N,MS)                                      DCLA   1
      DIMENSION A(1)                                                 DCLA   2
      DO 3 I=1,N                                                     DCLA   3
C        LOCATE DIAGONAL ELEMENT FOR ANY MATRIX STORAGE MODE         DCLA   4
      CALL LOC(I,I,ID,N,N,MS)                                        DCLA   5
C        REPLACE DIAGONAL ELEMENTS                                   DCLA   6
    3 A(ID)=C                                                        DCLA   7
      RETURN                                                         DCLA   8
      END                                                            DCLA   9
```

MSTR

Purpose:

Change storage mode of a matrix.

Usage:

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

Description of parameters:

- 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 - Same as MSA except for matrix R.

Remarks:

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

Matrix A must be a square matrix.

Subroutines and function subprograms required:

LOC

Method:

Matrix A is restructured 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.
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.

```
      SUBROUTINE MSTR(A,R,N,MSA,MSR)                                 MSTR   1
      DIMENSION A(1),R(1)                                            MSTR   2
      DO 20 I=1,N                                                    MSTR   3
      DO 20 J=1,N                                                    MSTR   4
C        IF R IS GENERAL, FORM ELEMENT                               MSTR   5
      IF(MSR) 5,10,5                                                 MSTR   6
C        IF IN LOWER TRIANGLE OF SYMMETRIC OR DIAGONAL R, BYPASS     MSTR   7
    5 IF(I-J) 10,10,20                                               MSTR   8
   10 CALL LOC(I,J,IR,N,N,MSR)                                       MSTR   9
C        IF IN UPPER AND OFF DIAGONAL  OF DIAGONAL R, BYPASS         MSTR  10
      IF(IR) 20,20,15                                                MSTR  11
C        OTHERWISE, FORM R(I,J)                                      MSTR  12
   15 R(IR)=0.0                                                      MSTR  13
      CALL LOC(I,J,IA,N,N,MSA)                                       MSTR  14
C        IF THERE IS NO A(I,J), LEAVE R(I,J) AT 0.0                  MSTR  15
      IF(IA) 20,20,18                                                MSTR  16
   18 R(IR)=A(IA)                                                    MSTR  17
   20 CONTINUE                                                       MSTR  18
      RETURN                                                         MSTR  19
      END                                                            MSTR  20
```

MFUN

Purpose:
Apply a function to each element of a matrix to form a resultant matrix.

Usage:

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

An external statement must precede call statement in order to identify parameter F as the name of a function.

Description of parameters:
- A - Name of input matrix.
- F - Name of FORTRAN-furnished or user function subprogram.
- 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:
Precision is dependent upon precision of function used.

Subroutines and function subprograms required:
LOC
F (see Description of Parameters)

Method:
Function F is applied to each element of matrix A to form matrix R.

```
      SUBROUTINE MFUN(A,F,R,N,M,MS)                                     MFUN   1
      DIMENSION A(1),R(1)                                               MFUN   2
C        COMPUTE VECTOR LENGTH, IT                                      MFUN   3
      CALL LOC(N,M,IT,N,M,MS)                                           MFUN   4
C        BUILD MATRIX R FOR ANY STORAGE MODE                            MFUN   5
      DO 5 I=1,IT                                                       MFUN   6
      B=A(I)                                                            MFUN   7
    5 R(I)=F(B)                                                         MFUN   8
      RETURN                                                            MFUN   9
      END                                                               MFUN  10
```

Function RECP

Purpose:
Calculate reciprocal of an element. This is a FORTRAN function subprogram which may be used as an argument by subroutine MFUN.

Usage:
RECP(E)

Description of parameters:
E - Matrix element.

Remarks:
Reciprocal of zero is taken to be 1.0E38.

Subroutines and function subprograms required:
None.

Method:
Reciprocal of element E is placed in RECP.

```
      FUNCTION RECP(E)                                                  RECP   1
      BIG=1.0E38                                                        RECP   2
C        TEST ELEMENT FOR ZERO                                          RECP   3
      IF(E) 1,2,1                                                       RECP   4
C        IF NON-ZERO, CALCULATE RECIPROCAL                              RECP   5
    1 RECP=1.0/E                                                        RECP   6
      RETURN                                                            RECP   7
C        IF ZERO, SET EQUAL TO INFINITY                                 RECP   8
    2 RECP=SIGN(BIG,E)                                                  RECP   9
      RETURN                                                            RECP  10
      END                                                               RECP  11
```

LOC

Purpose:
Compute a vector subscript for an element in a matrix of specified storage mode.

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

Description of parameters:
- 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.

Remarks:
None.

Subroutines and function subprograms required:
None.

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.

```
      SUBROUTINE LOC(I,J,IR,N,M,MS)                                     LOC     1
      IX=I                                                              LOC     2
      JX=J                                                              LOC     3
      IF(MS-1) 10,20,30                                                 LOC     4
   10 IRX=N*(JX-1)+IX                                                   LOC     5
      GO TO 36                                                          LOC     6
   20 IF(IX-JX) 22,24,24                                                LOC     7
   22 IRX=IX+(JX*JX-JX)/2                                               LOC     8
      GO TO 36                                                          LOC     9
   24 IRX=JX+(IX*IX-IX)/2                                               LOC    10
      GO TO 36                                                          LOC    11
   30 IRX=0                                                             LOC    12
      IF(IX-JX) 36,32,36                                                LOC    13
   32 IRX=IX                                                            LOC    14
   36 IR=IRX                                                            LOC    15
      RETURN                                                            LOC    16
      END                                                               LOC    17
```

ARRAY

Purpose:
Convert data array from single to double dimension or vice versa. This subroutine is used to link the user program which has double dimension arrays and the SSP subroutines which operate on arrays of data in a vector fashion.

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

Description of parameters:
- 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.
- S - If MODE=1, this vector contains, as input, a data matrix of size I by J in consecutive locations columnwise. If MODE=2, it contains a data matrix of the same size as output. The length of vector S is IJ, where IJ=I*J.
- D - If MODE=1, this matrix (N by M) contains, as output, a data matrix of size I by J in first I rows and J columns. If MODE=2, it contains a data matrix of the same size as input.

Remarks:
Vector S can be in the same location as matrix D. Vector S is referred as a matrix in other SSP routines, since it contains a data matrix. This subroutine converts only general data matrices (storage mode of 0).

Subroutines and function subroutines required:
None.

Method:
Refer to the discussion on variable data size in the section describing overall rules for usage in this manual.

```
      SUBROUTINE ARRAY (MODE,I,J,N,M,S,D)                              ARRAY  1
      DIMENSION S(1),D(1)                                              ARRAY  2
      NI=N-I                                                           ARRAY  3
C        TEST TYPE OF CONVERSION                                       ARRAY  4
      IF(MODE-1) 100, 100, 120                                         ARRAY  5
C        CONVERT FROM SINGLE TO DOUBLE DIMENSION                       ARRAY  6
  100 IJ=I*J+1                                                         ARRAY  7
      NM=N*J+1                                                         ARRAY  8
      DO 110 K=1,J                                                     ARRAY  9
      NM=NM-NI                                                         ARRAY 10
      DO 110 L=1,I                                                     ARRAY 11
      IJ=IJ-1                                                          ARRAY 12
      NM=NM-1                                                          ARRAY 13
  110 D(NM)=S(IJ)                                                      ARRAY 14
      GO TO 140                                                        ARRAY 15
C        CONVERT FROM DOUBLE TO SINGLE DIMENSION                       ARRAY 16
  120 IJ=0                                                             ARRAY 17
      NM=0                                                             ARRAY 18
      DO 130 K=1,J                                                     ARRAY 19
      DO 125 L=1,I                                                     ARRAY 20
      IJ=IJ+1                                                          ARRAY 21
      NM=NM+1                                                          ARRAY 22
  125 S(IJ)=D(NM)                                                      ARRAY 23
  130 NM=NM+NI                                                         ARRAY 24
  140 RETURN                                                           ARRAY 25
      END                                                              ARRAY 26
```

Mathematics — Integration and Differentiation

QSF

This subroutine performs the integration of an equidistantly tabulated function by Simpson's rule. To compute the vector of integral values:

$$z_i = z(x_i) = \int_a^{x_i} y(x)\,dx \quad \text{with } x_i = a + (i-1)\,h \quad \Big\} \; (i = 1,2,\ldots,n)$$

for a table of function values y_i $(i = 1,2,\ldots,n)$, given at equidistant points $x_i = a + (i-1)\,h$ $(i = 1,2,\ldots,n)$, Simpson's rule together with Newton's 3/8 rule or a combination of these two rules is used. Local truncation error is of the order h^5 in all cases with more than three points in the given table. Only z_2 has a truncation error of the order h^4 if there are only three points in the given table. No action takes place if the table consists of less than three sample points.

The function is assumed continuous and differentiable (three or four times, depending on the rule used).

Formulas used in this subroutine (z_j are integral values, y_j function values) are:

$$z_j = z_{j-1} + \frac{h}{3}(1.25\,y_{j-1} + 2y_j - 0.25\,y_{j+1}) \qquad (1)$$

$$z_j = z_{j-2} + \frac{h}{3}(y_{j-2} + 4y_{j-1} + y_j) \qquad (2)$$

(Simpson's rule)

$$z_j = z_{j-3} + \frac{3}{8}h\,(y_{j-3} + 3y_{j-2} + 3y_{j-1} + y_j) \qquad (3)$$

(Newton's 3/8 rule)

$$z_j = z_{j-5} + \frac{h}{3}(y_{j-5} + 3.875\,y_{j-4} + 2.625\,y_{j-3} + 2.625\,y_{j-2} + 3.875\,y_{j-1} + y_j) \qquad (4)$$

[combination of (2) and (3)]

Sometimes formula (2) is used in the following form:

$$z_j = z_{j+2} - \frac{h}{3}(y_j + 4\,y_{j+1} + y_{j+2}) \qquad (5)$$

Local truncation errors of formulas (1)...(4) are, respectively:

$$R_1 = \frac{1}{24}h^4 y'''(\xi_1) \quad (\xi_1 \in [x_{j-1}, x_{j+1}])$$

$$R_2 = -\frac{1}{90}h^5 y''''(\xi_2) \quad (\xi_2 \in [x_{j-2}, x_j])$$

$$R3 = -\frac{3}{80}h^5 y''''(\xi_3) \quad (\xi_3 \in [x_{j-3}, x_j])$$

$$R_4 = -\frac{1}{144}h^5 y''''(\xi_4) \quad (\xi_4 \in [x_{j-5}, x_j])$$

However, these truncation errors may accumulate.

For reference see:

(1) F.B. Hildebrand, Introduction to Numerical Analysis. McGraw-Hill, New York/Toronto/London, 1956, pp. 71-76.

(2) R. Zurmühl, Praktische Mathematik für Ingenieure und Physiker. Springer, Berlin/Göttingen/Heidelberg, 1963, pp. 214-221.

Subroutine QSF

Purpose:

To compute the vector of integral values for a given equidistant table of function values.

Usage:

CALL QSF(H,Y,Z,NDIM)

Description of parameters:

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

Remarks:

No action in case NDIM less then 3.

Subroutines and function subprograms required:

None

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 order H**5 (that is, fourth-order method). Only in case NDIM=3 truncation error of Z(2) is of order H**4.

```
      SUBROUTINE QSF(H,Y,Z,NDIM)                                        QSF  M01
      DIMENSION Y(1),Z(1)                                               QSF  M02
      HT=.3333333*H                                                     QSF  M03
      L1=1                                                              QSF  M04
      L2=2                                                              QSF  M05
      L3=3                                                              QSF  M06
      L4=4                                                              QSF  M07
      L5=5                                                              QSF  M08
      L6=6                                                              QSF  M09
      IF(NDIM-5)7,8,1                                                   QSF  M10
C     NDIM IS GREATER THAN 5. PREPARATIONS OF INTEGRATION LOOP          QSF  M11
    1 SUM1=Y(L2)+Y(L2)                                                  QSF  M12
      SUM1=SUM1+SUM1                                                    QSF  M13
      SUM1=HT*(Y(L1)+SUM1+Y(L3))                                        QSF  M14
      AUX1=Y(L4)+Y(L4)                                                  QSF  M15
      AUX1=AUX1+AUX1                                                    QSF  M16
      AUX1=SUM1+HT*(Y(L3)+AUX1+Y(L5))                                   QSF  M17
      AUX2=HT*(Y(L1)+3.875*(Y(L2)+Y(L5))+2.625*(Y(L3)+Y(L4))+Y(L6))     QSF  M18
      SUM2=Y(L5)+Y(L5)                                                  QSF  M19
      SUM2=SUM2+SUM2                                                    QSF  M20
      SUM2=AUX2-HT*(Y(L4)+SUM2+Y(L6))                                   QSF  M21
      Z(L1)=0.                                                          QSF  M22
      AUX=Y(L3)+Y(L3)                                                   QSF  M23
      AUX=AUX+AUX                                                       QSF  M24
      Z(L2)=SUM2-HT*(Y(L2)+AUX+Y(L4))                                   QSF  M25
      Z(L3)=SUM1                                                        QSF  M26
      Z(L4)=SUM2                                                        QSF  M27
      IF(NDIM-6)5,5,2                                                   QSF  M28
C     INTEGRATION LOOP                                                  QSF  M29
    2 DO 4 I=7,NDIM,2                                                   QSF  M30
      SUM1=AUX1                                                         QSF  M31
      SUM2=AUX2                                                         QSF  M32
      AUX1=Y(I-1)+Y(I-1)                                                QSF  M33
      AUX1=AUX1+AUX1                                                    QSF  M34
      AUX1=SUM1+HT*(Y(I-2)+AUX1+Y(I))                                   QSF  M35
      Z(I-2)=SUM1                                                       QSF  M36
      IF(I-NDIM)3,6,6                                                   QSF  M37
    3 AUX2=Y(I)+Y(I)                                                    QSF  M38
      AUX2=AUX2+AUX2                                                    QSF  M39
      AUX2=SUM2+HT*(Y(I-1)+AUX2+Y(I+1))                                 QSF  M40
    4 Z(I-1)=SUM2                                                       QSF  M41
    5 Z(NDIM-1)=AUX1                                                    QSF  M42
      Z(NDIM)=AUX2                                                      QSF  M43
      RETURN                                                            QSF  M44
    6 Z(NDIM-1)=SUM2                                                    QSF  M45
      Z(NDIM)=AUX1                                                      QSF  M46
      RETURN                                                            QSF  M47
C     END OF INTEGRATION LOOP                                           QSF  M48
    7 IF(NDIM-3)12,11,8                                                 QSF  M49
C     NDIM IS EQUAL TO 4 OR 5                                           QSF  M50
    8 SUM2=1.125*HT*(Y(L1)+Y(L2)+Y(L2)+Y(L2)+Y(L3)+Y(L3)+Y(L3)+Y(L4))   QSF  M51
      SUM1=Y(L2)+Y(L2)                                                  QSF  M52
      SUM1=SUM1+SUM1                                                    QSF  M53
      SUM1=HT*(Y(L1)+SUM1+Y(L3))                                        QSF  M54
      Z(L1)=0.                                                          QSF  M55
      AUX1=Y(L3)+Y(L3)                                                  QSF  M56
      AUX1=AUX1+AUX1                                                    QSF  M57
      Z(L2)=SUM2-HT*(Y(L2)+AUX1+Y(L4))                                  QSF  M58
      IF(NDIM-5)10,9,9                                                  QSF  M59
    9 AUX1=Y(L4)+Y(L4)                                                  QSF  M60
      AUX1=AUX1+AUX1                                                    QSF  M61
      Z(L5)=SUM1+HT*(Y(L3)+AUX1+Y(L5))                                  QSF  M62
   10 Z(L3)=SUM1                                                        QSF  M63
      Z(L4)=SUM2                                                        QSF  M64
      RETURN                                                            QSF  M65
C     NDIM IS EQUAL TO 3                                                QSF  M66
   11 SUM1=HT*(1.25*Y(L1)+Y(L2)+Y(L2)-.25*Y(L3))                        QSF  M67
      SUM2=Y(L2)+Y(L2)                                                  QSF  M68
      SUM2=SUM2+SUM2                                                    QSF  M69
      Z(L3)=HT*(Y(L1)+SUM2+Y(L3))                                       QSF  M70
      Z(L1)=0.                                                          QSF  M71
      Z(L2)=SUM1                                                        QSF  M72
   12 RETURN                                                            QSF  M73
      END                                                               QSF  M74
```

This subroutine performs the integration of a given function by the trapezoidal rule together with Romberg's extrapolation method in order to compute an approximation for:

$$y = \int_a^b f(x)\,dx \qquad (1)$$

Successively dividing the interval $[a,b]$ into 2^i equidistant subintervals ($i = 0, 1, 2, \ldots$) and using the following notations:

$$h_i = \frac{b-a}{2^i}\ ;\ x_{i,k} = a + k \cdot h_i,\ f_{i,k} = f(x_{i,k})$$

$$(k = 0, 1, 2, \ldots, 2^i)$$

the trapezoidal rule gives approximations $T_{o,i}$ to the integral value y:

$$T_{o,i} = h_i \left\{ \sum_{k=0}^{2^i} f_{i,k} - \frac{1}{2}\,(f(a) + f(b)) \right\} \qquad (2)$$

Then the following can be written:

$$T_{o,i} = y + \sum_{r=1}^{\infty} C_{o,2r} \cdot h_i^{2r}$$

with unknown coefficients $C_{o,2r}$ which do not depend on i. Thus there is a truncation error of the order h_i^2.

Knowing two successive approximations, $T_{o,i}$ and $T_{o,i+1}$, an extrapolated value can be generated:

$$T_{1,i} = T_{o,\,i+1} + \frac{T_{o,i+1} - T_{o,i}}{2^2 - 1} \qquad (3)$$

This is a better approximation to y because:

$$T_{1,i} = y + \frac{1}{2^2-1} \sum_{r=1}^{\infty} C_{o,2r}\,(2^2\, h_{i+1}^{2r} - h_i^{2r})$$

Noting that $2^2\, h_{i+1}^2 - h_i^2 = 0$ and setting:

$$C_{1,2r} = \frac{1}{2^2-1}\,(2^2 - 2^{2r}) \cdot C_{o,2r}$$

$T_{1,i}$ becomes:

$$T_{1,i} = y + \sum_{r=2}^{\infty} C_{1,2r}\, h_{i+1}^{2r}$$

This gives a truncation error of the order h_{i+1}^4.

Knowing $T_{o,\,i+2}$ also, $T_{1,\,i+1}$ can be generated (formula 3), and:

$$T_{2,i} = T_{1,\,i+1} + \frac{T_{1,i+1} - T_{1,i}}{2^4 - 1} \qquad (4)$$

Thus:

$$T_{2,i} = y + \sum_{r=3}^{\infty} C_{2,2r} \cdot h_{i+2}^{2r}$$

with $C_{2,2r} = \frac{1}{2^4-1}\,(2^4 - 2^{2r})\, C_{1,2r}$

with a truncation error of the order h_{i+2}^6. Observe that the order of truncation error increases by 2 at each new extrapolation step.

The subroutine uses the scheme shown in the figure below for computation of T-values and

generates the upward diagonal in the one-dimensional storage array AUX, using the general formula:

$$T_{k,j} = T_{k-1,j+1} + \frac{T_{k-1,j+1} - T_{k-1,j}}{2^{2k}-1} \quad (k+j=i, \quad j = i-1, i-2, \ldots, 2, 1, 0) \tag{5}$$

and storing:

$T_{0,i}$ into AUX (i+1)

$T_{1,i-1}$ into AUX (i)

....

$T_{k,0}$ into AUX (1)

Truncation error		$O(h_i^2)$	$O(h_i^4)$	$O(h_i^6)$	$O(h_i^8)\ldots$
step length h_i	i \ j	0	1	2	3 ...
b-a	0	$T_{0,0}$	$T_{1,0}$	$T_{2,0}$	$T_{3,0}$ ...
$\frac{b-a}{2}$	1	$T_{0,1}$	$T_{1,1}$	$T_{2,1}$	⋮
$\frac{b-a}{4}$	2	$T_{0,2}$	$T_{1,2}$	⋮	
$\frac{b-a}{8}$	3	$T_{0,3}$	⋮		
⋮	⋮	⋮			

Computation of T-values (QATR)

The procedure stops if the difference between two successive values of AUX (1) is less than a given tolerance, or if the values of AUX (1) start oscillating, thus showing the influence of rounding errors.

Subroutine QATR

Purpose:

To compute an approximation for integral (FCT(X), summed over X from XL to XU).

Usage:

CALL QATR(XL, XU, EPS, NDIM, FCT, Y, IER, AUX) Parameter FCT required an EXTERNAL statement.

Description of parameters:

XL - The lower bound of the interval.

XU - The upper bound of the interval.

EPS - The upper bound of the absolute error.

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 - A resulting error parameter.

AUX - An auxiliary storage array with dimension NDIM.

Remarks:

Error parameter IER is coded in the following form:

IER=0 - It was possible to reach the required accuracy. No error.

IER=1 - It is impossible to reach the required accuracy because of rounding errors.

IER=2 - It was impossible to check accuracy because NDIM is less than 5, or the required accuracy could not be reached within NDIM-1 steps. NDIM should be increased.

Subroutines and function subprograms required:

The external function subprogram FCT(X) must be coded by the user. Its argument X should not be destroyed.

Method:

Evaluation of Y is done by means of the trapezoidal rule in connection with Romberg's principle. On return Y contains the best possible approximation of the integral value and vector AUX the upward diagonal of the Romberg scheme. Components AUX(I) (I=1, 2,..., IEND, with IEND less than or equal to NDIM) become approximation to the integral value with decreasing accuracy by multiplication by (XU-XL).

For reference see:

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

2. Bauer, Algorithm 60, CACM, Vol. 4, Iss. 6 (1961), pp. 255.


```
      SUBROUTINE QATR(XL,XU,EPS,NDIM,FCT,Y,IER,AUX)                    QATR   1
      DIMENSION AUX(1)                                                 QATR   2
C     PREPARATIONS OF ROMBERG-LOOP                                     QATR   3
      AUX(1)=.5*(FCT(XL)+FCT(XU))                                      QATR   4
      H=XU-XL                                                          QATR   5
      IF(NDIM-1)18,8,1                                                 QATR   6
    1 IF(H)2,10,2                                                      QATR   7
C     NDIM IS GREATER THAN 1 AND H IS NOT EQUAL TO 0.                  QATR   8
    2 HH=H                                                             QATR   9
      E=EPS/ABS(H)                                                     QATR  10
      DELT2=0.                                                         QATR  11
      P=1.                                                             QATR  12
      JJ=1                                                             QATR  13
      DO 7 I=2,NDIM                                                    QATR  14
      Y=AUX(1)                                                         QATR  15
      DELT1=DELT2                                                      QATR  16
      HD=HH                                                            QATR  17
      HH=.5*HH                                                         QATR  18
      P=.5*P                                                           QATR  19
      X=XL+HH                                                          QATR  20
      SW=0.                                                            QATR  21
```

```
      DO 3 J=1,JJ                                                    QATR  22
      SM=SM+FCT(X)                                                   QATR  23
    3 X=X+HD                                                         QATR  24
      AUX(I)=.5*AUX(I-1)+P*SM                                        QATR  25
C     A NEW APPROXIMATION OF INTEGRAL VALUE IS COMPUTED BY MEANS OF  QATR  26
C     TRAPEZOIDAL RULE.                                              QATR  27
C     START OF ROMBERGS EXTRAPOLATION METHOD.                        QATR  28
      Q=1.                                                           QATR  29
      JI=I-1                                                         QATR  30
      DO 4 J=1,JI                                                    QATR  31
      II=I-J                                                         QATR  32
      Q=Q+Q                                                          QATR  33
      Q=Q+Q                                                          QATR  34
    4 AUX(II)=AUX(II+1)+(AUX(II+1)-AUX(II))/(Q-1.)                   QATR  35
C     END OF ROMBERG-STEP                                            QATR  36
      DELT2=ABS(Y-AUX(1))                                            QATR  37
      IF(I-9)17,5,5                                                  QATR  38
    5 IF(DELT2-E)10,10,6                                             QATR  39
    6 IF(DELT2-DELT1)7,11,11                                         QATR  40
    7 JJ=JJ+JJ                                                       QATR  41
    8 IER=2                                                          QATR  42
    9 Y=H*AUX(1)                                                     QATR  43
      RETURN                                                         QATR  44
   10 IER=0                                                          QATR  45
      GO TO 9                                                        QATR  46
   11 IER=1                                                          QATR  47
      Y=H*Y                                                          QATR  48
      RETURN                                                         QATR  49
      END                                                            QATR  50
```

Mathematics - Ordinary Differential Equations

RK1

This subroutine integrates a given function using the Runge-Kutta technique and produces the final computed value of the integral.

The ordinary differential equation:

$$\frac{dy}{dx} = f(x, y) \tag{1}$$

with initial condition $y(x_0) = y_0$ is solved numerically using a fourth-order Runge-Kutta integration process. This is a single-step method in which the value of y at $x = x_n$ is used to compute $y_{n+1} = y(x_{n+1})$ and earlier values y_{n-1}, y_{n-2}, etc., are not used.

The relevant formulae are:

$$y_{n+1} = y_n + 1/6\,[k_0 + 2k_1 + 2k_2 + k_3] \tag{2}$$

where we define, for step size h

$$\begin{cases} k_0 = hf(x_n, y_n) \\ k_1 = hf(x_n + h/2,\ y_n + k_0/2) \\ k_2 = hf(x_n + h/2,\ y_n + k_1/2) \\ k_3 = hf(x_n + h,\ y_n + k_2) \end{cases} \tag{3}$$

Subroutine RK1

Purpose:

Integrates a first order differential equation DY/DX= FUN(X, Y) up to a specified final value.

Usage:

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

Description of parameters:

- FUN - User-supplied function subprogram with arguments X, Y which gives 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 - Error code:
  - IER=0 No error.
  - IER=1 Step size is zero.

Remarks:

If XI is greater than XF, ANSX=XI and ANSY=YI.

If H is zero, IER is set to one, ANSX is set to XI, and ANSY is set to zero.

Subroutines and function subprograms required:

FUN is a two argument function subprogram furnished by the user: DY/DX= FUN (X, Y). Calling program must have FORTRAN external statement containing names of function subprograms listed in call to RK1.

Method:

Uses fourth-order Runge-Kutta integration process on a recursive basis as shown in F. B. Hildebrand, 'Introduction to Numerical Analysis', McGraw-Hill, 1956. Process is terminated and final value adjusted when either XF or YF is reached.

```
      SUBROUTINE RK1(FUN,HI,XI,YI,XF,YF,ANSX,ANSY,IER)               RK1   1
C        IF XF IS LESS THAN OR EQUAL TO XI, RETURN XI,YI AS ANSWER   RK1   2
      IF(XF-XI) 11,11,12                                             RK1   3
   11 ANSX=XI                                                        RK1   4
      ANSY=YI                                                        RK1   5
      RETURN                                                         RK1   6
C        TEST INTERVAL VALUE                                         RK1   7
   12 H=HI                                                           RK1   8
      IF(HI) 16,14,20                                                RK1   9
   14 IER=1                                                          RK1  10
      ANSX=XI                                                        RK1  11
      ANSY=0.0                                                       RK1  12
      RETURN                                                         RK1  13
   16 H=-HI                                                          RK1  14
C        SET XN=INITIAL X,YN=INITIAL Y                               RK1  15
   20 XN=XI                                                          RK1  16
      YN=YI                                                          RK1  17
C        INTEGRATE ONE TIME STEP                                     RK1  18
      HNEW=H                                                         RK1  19
      JUMP=1                                                         RK1  20
      GO TO 170                                                      RK1  21
   25 XN1=XX                                                         RK1  22
      YN1=YY                                                         RK1  23
C        COMPARE XN1 (=X(N+1)) TO X FINAL AND BRANCH ACCORDINGLY     RK1  24
      IF(XN1-XF)50,30,40                                             RK1  25
C        XN1=XF, RETURN (XF,YN1) AS ANSWER                           RK1  26
   30 ANSX=XF                                                        RK1  27
      ANSY=YN1                                                       RK1  28
      GO TO 160                                                      RK1  29
C        XN1 GREATER THAN XF, SET NEW STEP SIZE AND INTEGRATE ONE STEP RK1  30
C        RETURN RESULTS OF INTEGRATION AS ANSWER                     RK1  31
   40 HNEW=XF-XN                                                     RK1  32
      JUMP=2                                                         RK1  33
      GO TO 170                                                      RK1  34
   45 ANSX=XX                                                        RK1  35
      ANSY=YY                                                        RK1  36
      GO TO 160                                                      RK1  37
C        XN1 LESS THAN X FINAL, CHECK IF (YN,YN1) SPAN Y FINAL       RK1  38
   50 IF((YN1-YF)*(YF-YN))60,70,110                                  RK1  39
```

```
C        YN1 AND YN DO NOT SPAN YF. SET (XN,YN) AS (XN1,YN1) AND REPEAT RK1   40
   60 YN=YN1                                                          RK1   41
      XN=XN1                                                          RK1   42
      GO TO 170                                                       RK1   43
C        EITHER YN OR YN1 =YF. CHECK WHICH AND SET PROPER (X,Y) AS ANSWERK1   44
   70 IF(YN1-YF)80,100,80                                             RK1   45
   80 ANSY=YN                                                         RK1   46
      ANSX=XN                                                         RK1   47
      GO TO 160                                                       RK1   48
  100 ANSY=YN1                                                        RK1   49
      ANSX=XN1                                                        RK1   50
      GO TO 160                                                       RK1   51
C        YN AND YN1 SPAN YF. TRY TO FIND X VALUE ASSOCIATED WITH YF    RK1   52
  110 DO 140 I=1,10                                                   RK1   53
C        INTERPOLATE TO FIND NEW TIME STEP AND INTEGRATE ONE STEP      RK1   54
C        TRY TEN INTERPOLATIONS AT MOST                                RK1   55
      HNEW=(YF-YN )/(YN1-YN))*(XN1-XN)                                RK1   56
      JUMP=3                                                          RK1   57
      GO TO 170                                                       RK1   58
  115 XNEW=XX                                                         RK1   59
      YNEW=YY                                                         RK1   60
C        COMPARE COMPUTED Y VALUE WITH YF AND BRANCH                   RK1   61
      IF(YNEW-YF)120,150,130                                          RK1   62
C        ADVANCE, YF IS BETWEEN YNEW AND YN1                           RK1   63
  120 YN=YNEW                                                         RK1   64
      XN=XNEW                                                         RK1   65
      GO TO 140                                                       RK1   66
C        ADVANCE, YF IS BETWEEN YN AND YNEW                            RK1   67
  130 YN1=YNEW                                                        RK1   68
      XN1=XNEW                                                        RK1   69
  140 CONTINUE                                                        RK1   70
C        RETURN (XNEW,YF) AS ANSWER                                    RK1   71
  150 ANSX=XNEW                                                       RK1   72
      ANSY=YF                                                         RK1   73
  160 RETURN                                                          RK1   74
  170 H2=HNEW/2.0                                                     RK1   75
      T1=HNEW*FUN(XN,YN)                                              RK1   76
      T2=HNEW*FUN(XN+H2,YN+T1/2.0)                                    RK1   77
      T3=HNEW*FUN(XN+H2,YN+T2/2.0)                                    RK1   78
      T4=HNEW*FUN(XN+HNEW,YN+T3)                                      RK1   79
      YY=YN+(T1+2.0*T2+2.0*T3+T4)/6.0                                 RK1   80
      XX=XN+HNEW                                                      RK1   81
      GO TO (25,45,115), JUMP                                         RK1   82
      END                                                             RK1   83
```

RK2

This subroutine integrates a given function using the Runge-Kutta technique and produces tabulated values of the computed integral.

The ordinary differential equation:

$$\frac{dy}{dx} = f(x, y) \tag{1}$$

with initial condition $y(x_0) = y_0$ is solved numerically using a fourth-order Runge-Kutta integration process. This is a single-step method in which the value of y at $x = x_n$ is used to compute $y_{n+1} = y(x_{n+1})$ and earlier values y_{n-1}, y_{n-2}, etc., are not used.

The relevant formulae are:

$$y_{n+1} = y_n + 1/6\left[k_0 + 2k_1 + 2k_2 + k_3\right] \tag{2}$$

where we define, for step size h

$$\begin{cases} k_0 = hf(x_n, y_n) \\ k_1 = hf(x_n + h/2, y_n + k_0/2) \\ k_2 = hf(x_n + h/2, y_n + k_1/2) \\ k_3 = hf(x_n + h, y_n + k_2) \end{cases} \tag{3}$$

Subroutine RK2

Purpose:

Integrates a first-order differential equation DY/DX=FUN(X, Y) and produces a table of integrated values.

Usage:

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

Description of parameters:

- FUN - User-supplied function subprogram with arguments X, Y which gives 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.

Remarks:

None.

Subroutines and function subprograms required:

FUN - User-supplied function subprogram for DY/DX.

Calling program must have FORTRAN EXTERNAL statement containing names of function subprograms listed in call to RK2.

Method:

Fourth-order Runge-Kutta integration on a recursive basis as shown in F. B. Hildebrand, 'Introduction to Numerical Analysis', McGraw-Hill, New York, 1956.

```
      SUBROUTINE RK2(FUN,H,XI,YI,K,N,VEC)                             RK2    1
      DIMENSION VEC(1)                                                RK2    2
      H2=H/2.                                                         RK2    3
      Y=YI                                                            RK2    4
      X=XI                                                            RK2    5
      DO 2 I=1,N                                                      RK2    6
      DO 1 J=1,K                                                      RK2    7
      T1=H*FUN(X,Y)                                                   RK2    8
      T2=H*FUN(X+H2,Y+T1/2.)                                          RK2    9
      T3=H*FUN(X+H2,Y+T2/2.)                                          RK2   10
      T4=H*FUN(X+H,Y+T3)                                              RK2   11
      Y= Y+(T1+2.*T2+2.*T3+T4)/6.                                     RK2   12
    1 X=X+H                                                           RK2   13
    2 VEC(I)=Y                                                        RK2   14
      RETURN                                                          RK2   15
      END                                                             RK2   16
```

RKGS

This subroutine uses the Runge-Kutta method for the solution of initial-value problems.

The purpose of the Runge-Kutta method is to obtain an approximate solution of a system of first-order ordinary differential equations with given initial values. It is a fourth-order integration procedure which is stable and self-starting; that is, only the functional values at a single previous point are required to obtain the functional values ahead. For this reason it is easy to change the step size h at any step in the calculations. On the other hand, each Runge-Kutta step requires the evaluation of the right-hand side of the system four times, which is a great disadvantage compared with other methods of the same order of accuracy, especially predictor-corrector methods. Another disadvantage of the method is that neither the truncation errors nor estimates of them are obtained in the calculation procedure. Therefore, control of accuracy and adjustment of the step size h is done by comparison of the results due to double and single step size 2h and h.

Given the system of first-order ordinary differential equations:

$$y_1' = \frac{dy_1}{dx} = f_1(x, y_1, y_2, \ldots, y_n)$$

$$y_2' = \frac{dy_2}{dx} = f_2(x, y_1, y_2, \ldots, y_n)$$

$$\ldots\ldots\ldots\ldots$$

$$y_n' = \frac{dy_n}{dx} = f_n(x, y_1, y_2, \ldots, y_n)$$

and the initial values:

$$y_1(x_0) = y_{1,0},\ y_2(x_0) = y_{2,0},\ \ldots,\ y_n(x_0) = y_{n,0}$$

and using the following vector notations:

$$Y(x) = \begin{pmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_n(x) \end{pmatrix},\quad F(x, Y) = \begin{pmatrix} f_1(x, Y) \\ f_2(x, Y) \\ \vdots \\ f_n(x, Y) \end{pmatrix},\quad Y_0 = \begin{pmatrix} y_{1,0} \\ y_{2,0} \\ \vdots \\ y_{n,0} \end{pmatrix}$$

where Y, F and Y_0 are column vectors, the given problem appears as follows:

$$Y' = \frac{dY}{dx} = F(x, Y) \text{ with } Y(x_0) = Y_0$$

With respect to storage requirements and compensation of accumulated roundoff errors, Gill's modification of the classical Runge-Kutta formulas is preferred. Thus, starting at x_0 with $Y(x_0) = Y_0$ and vector $Q_0 = 0$, the resulting vector $Y4 = Y(x_0 + h)$ is computed by the following formulas:

$$K_1 = hF(x_0, Y_0) \quad ; Y_1 = Y_0 + \frac{1}{2}(K_1 - 2Q_0)$$

$$Q_1 = Q_0 + 3\left[\frac{1}{2}(K_1 - 2Q_0)\right] - \frac{1}{2}K_1$$

$$K_2 = hF(x_0 + \frac{h}{2}, Y_1) \ ; Y_2 = Y_1 + (1 - \sqrt{\tfrac{1}{2}})(K_2 - Q_1)$$

$$Q_2 = Q_1 + 3\left[(1 - \sqrt{\tfrac{1}{2}})(K_2 - Q_1)\right] - (1 - \sqrt{\tfrac{1}{2}})K_2 \qquad (1)$$

$$K_3 = hF(x_0 + \frac{h}{2}, Y_2) \ ; Y_3 = Y_2 + (1 + \sqrt{\tfrac{1}{2}})(K_3 - Q_2)$$

$$Q_3 = Q_2 + 3\left[(1 + \sqrt{\tfrac{1}{2}})(K_3 - Q_2)\right] - (1 + \sqrt{\tfrac{1}{2}})K_3$$

$$K_4 = hF(x_0 + h, Y_3) \ ; Y_4 = Y_3 + \frac{1}{6}(K_4 - 2Q_3)$$

$$Q_4 = Q_3 + 3\left[\frac{1}{6}(K_4 - 2Q_3)\right] - \frac{1}{2}K_4$$

where K_1, K_2, K_3, K_4, Y_1, Y_2, Y_3, Y_4, Q_1, Q_2, Q_3, Q_4 are all column vectors with n components. If the procedure were carried out with infinite precision (that is, no rounding errors), vector Q_4 defined above would be zero. In practice this is not true, and Q_4 represents approximately three times the roundoff error in Y_4 accumulated during one step. To compensate for this accumulated roundoff, Q_4 is used as Q_0 for the next step. Also $(x_0 + h)$ and Y_4 serve as x_0 and Y_0 respectively at the next step.

For initial control of accuracy, an approximation for $Y(x_0 + 2h)$ called $Y^{(2)}(x_0 + 2h)$ is computed using the step size 2h, and then an approximation called $Y^{(1)}(x_0 + 2h)$, using two times the step size h. From these two approximations, a test value δ for accuracy is generated in the following way:

$$\delta = \frac{1}{15}\sum_{i=1}^{n} a_i \cdot \left| y_i^{(1)} - y_i^{(2)} \right| \qquad (2)$$

where the coefficients ai are error-weights specified in the input of the procedure.

Test value δ is an approximate measure for the local truncation error at point x_0+2h. If δ is greater than a given tolerance ϵ_2, increment h is halved and the procedure starts again at the point x_0. If δ is less than ϵ_2, the results $Y^{(1)}(x_0+h)$ and $Y^{(1)}(x_0+2h)$

are assumed to be correct. They are then handed, together with $x_0 + h$ and $x_0 + 2h$ and the derivatives at these points -- that is, the values of $F[x_0 + h, Y^{(1)}(x_0+h)]$ and $F[x_0+2h, Y^{(1)}(x_0+2h)]$ respectively -- to a user-supplied output subroutine.

If δ is less than $\epsilon_1 = \epsilon_2/50$, the next step is carried out with the doubled increment. However, care is taken in the procedure that the increment never becomes greater than the increment h specified as an input parameter, and further that all points $x_0 + jh$ (where $j = 1, 2, \ldots$) which are situated between the lower and upper bound of the integration interval are included in the output. Finally, the increment of the last step of the procedure is chosen in such a way that the upper bound of the integration interval is reached exactly.

The entire input of the procedure is:

1. Lower and upper bound of the integration interval, initial increment of the independent variable, upper bound ϵ_2 of the local truncation error.
2. Initial values of the dependent variables and weights for the local truncation errors in each component of the dependent variables.
3. The number of differential equations in the system.
4. As external subroutine subprograms, the computation of the right-hand side of the system of differential equations; for flexibility in output, an output subroutine.
5. An auxiliary storage array named AUX with 8 rows and n columns.

Output is done in the following way. If a set of approximations to the dependent variables Y(x) is found to be of sufficient accuracy, it is handed -- together with x, the derivative $F[x, Y(x)]$, the number of bisections of the initial increment, the number of differential equations, the lower and upper bound of the interval, the initial step size, error bound ϵ_2, and a parameter for terminating subroutine RKGS -- to the output subroutine. Because of this output subroutine, the user has the opportunity to choose his own output format, to handle the output values as he wants, to change the upper error bound, and to terminate subroutine RKGS at any output point. In particular, the user is able to drop the output of some intermediate points, printing only the result values at the special points $x_0 + nh$ ($n = 0, 1, 2, \ldots$). The user may also perform intermediate computation using the integration results before continuing the process.

For better understanding of the flowchart and of the FORTRAN program, the following figure shows the allocation of special intermediate result vectors within the storage array AUX.

For reference see A. Ralston/H. S. Wilf, Mathematical Methods for Digital Computers , Wiley, New York/London, 1960, pp. 110-120.

AUX

function vector Y(x)	1. row (AUX (1) in flowchart)
derivative vector F(x, Y(x))	2. row (AUX (2) in flowchart)
vector of accumulated roundoff at point x	3. row (AUX (3) in flowchart)
function vector Y(x+2h) for testing purposes	4. row (AUX (4) in flowchart)
function vector Y(x+h)	5. row (AUX (5) in flowchart)
vector of accumulated roundoff at point x+h	6. row (AUX (6) in flowchart)
derivative vector F(x+h, Y(x+h))	7. row (AUX (7) in flowchart)
vector of error weights multiplied by 1/15	8. row (AUX (8) in flowchart)

Storage allocation in auxiliary storage array AUX (RKGS)

Subroutine RKGS

Purpose:

To solve a system of first-order ordinary differential equations with given initial values.

Usage:

CALL RKGS(PRMT, Y, DERY, NDIM, IHLF, FCT, OUTP, AUX) Parameters FCT and OUTP require an external statement.

Description of parameters:

PRMT - An input and output vector with dimension greater than or equal to 5, which specifies the parameters of the interval and of accuracy and which serves for communication between the output subroutine (furnished by the user) and subroutine RKGS. Except for PRMT(5), the components are not destroyed by subroutine RKGS and they 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), the increment gets halved. If the increment is less than PRMT(3) and absolute error less than PRMT(4)/50, the increment gets doubled. The user may change PRMT(4) in his output subroutine.

PRMT(5) - No input parameter. Subroutine RKGS initializes PRMT(5)=0. If the user wants to terminate subroutine RKGS at any output point, he must change PRMT(5) to nonzero in subroutine OUTP. Further components of vector PRMT can be made available if its dimension is defined greater than 5. However subroutine RKGS does not require this. Nevertheless, they may be useful for handling result values to the main program (calling RKGS) which are obtained by special manipulations with output data in subroutine OUTP.

Y - Input vector of initial values (destroyed). On return, Y is the resultant vector of dependent variables computed at intermediate points X.

DERY - Input vector of error weights (destroyed). The sum of its components must equal 1. On return, DERY is the vector of derivatives of function values Y at points X.

NDIM - An input value which specifies the number of equations in the system.

IHLF - An output value which specifies the number of bisections of the initial increment. When IHLF is greater than 10, subroutine RKGS exits to the main program with error message IHLF=11. Other error messages are:

IHLF=12; PRMT(3)=0 or PRMT(1)=PRMT(2)

IHLF=13; SIGN(PRMT(3)) is not equal to SIGN(PRMT(2)-PRMT(1)).

FCT - The name of the external subroutine used. This subroutine computes the right-hand side, DERY, of the system for 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 the 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 8 rows and NDIM columns.

Remarks:

The procedure terminates and returns to the calling program, if

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

Subroutines and function subprograms required:

The external subroutines FCT(X, Y, DERY) and OUTP(X, Y, DERY, IHLF, NDIM, PRMT) must be furnished by the user.

Method:

Evaluation is done by means of fourth-order Runge-Kutta formulae using the modification due to Gill. Accuracy is tested comparing the results of the procedure with the increment.

Subroutine 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, the subroutine returns with error message IHLF=11 to the main program.

To get full flexibility in output, an output subroutine must be furnished by the user.

```
      SUBROUTINE RKGS(PRMT,Y,DERY,NDIM,IHLF,FCT,OUTP,AUX)                RKGS   1
      DIMENSION Y(1),DERY(1),AUX(8,1),A(4),B(4),C(4),PRMT(5)             RKGS M01
      DO 1 I=1,NDIM                                                      RKGS   3
    1 AUX(8,I)=.06666667*DERY(I)                                         RKGS   4
      X=PRMT(1)                                                          RKGS   5
      XEND=PRMT(2)                                                       RKGS   6
      H=PRMT(3)                                                          RKGS   7
      PRMT(5)=0.                                                         RKGS   8
      CALL FCT(X,Y,DERY)                                                 RKGS   9
C     ERROR TEST                                                         RKGS  10
      IF(H*(XEND-X))38,37,2                                              RKGS  11
C     PREPARATIONS FOR RUNGE-KUTTA METHOD                                RKGS  12
    2 A(1)=.5                                                            RKGS  13
      A(2)=.2928932                                                      RKGS  14
      A(3)=1.707107                                                      RKGS  15
      A(4)=.1666667                                                      RKGS  16
      B(1)=2.                                                            RKGS  17
      B(2)=1.                                                            RKGS  18
      B(3)=1.                                                            RKGS  19
      B(4)=2.                                                            RKGS  20
      C(1)=.5                                                            RKGS  21
      C(2)=.2928932                                                      RKGS  22
      C(3)=1.707107                                                      RKGS  23
      C(4)=.5                                                            RKGS  24
C     PREPARATIONS OF FIRST RUNGE-KUTTA STEP                             RKGS  25
      DO 3 I=1,NDIM                                                      RKGS  26
      AUX(1,I)=Y(I)                                                      RKGS  27
      AUX(2,I)=DERY(I)                                                   RKGS  28
      AUX(3,I)=0.                                                        RKGS  29
    3 AUX(4,I)=0.                                                        RKGS  30
      IREC=0                                                             RKGS  31
      H=H+H                                                              RKGS  32
      IHLF=-1                                                            RKGS  33
      ISTEP=0                                                            RKGS  34
      IEND=0                                                             RKGS  35
C     START OF A RUNGE-KUTTA STEP                                        RKGS  36
    4 IF((X+H-XEND)*H)7,6,5                                              RKGS  37
    5 H=XEND-X                                                           RKGS  38
    6 IEND=1                                                             RKGS  39
C     RECORDING OF INITIAL VALUES OF THIS STEP                           RKGS  40
    7 CALL OUTP(X,Y,DERY,IREC,NDIM,PRMT)                                 RKGS  41
      IF(PRMT(5))40,8,40                                                 RKGS  42
    8 ITEST=0                                                            RKGS  43
    9 ISTEP=ISTEP+1                                                      RKGS  44
C     START OF INNERMOST RUNGE-KUTTA LOOP                                RKGS  45
      J=1                                                                RKGS  46
   10 AJ=A(J)                                                            RKGS  47
      BJ=B(J)                                                            RKGS  48
      CJ=C(J)                                                            RKGS  49
      DO 11 I=1,NDIM                                                     RKGS  50
```

```
      R1=H*DERY(I)                                                      RKGS  51
      R2=AJ*(R1-BJ*AUX(6,I))                                            RKGS  52
      Y(I)=Y(I)+R2                                                      RKGS  53
      R2=R2+R2+R2                                                       RKGS  54
   11 AUX(6,I)=AUX(6,I)+R2-CJ*R1                                        RKGS  55
      IF(J-4)12,15,15                                                   RKGS  56
   12 J=J+1                                                             RKGS  57
      IF(J-3)13,14,13                                                   RKGS  58
   13 X=X+.5*H                                                          RKGS  59
   14 CALL FCT(X,Y,DERY)                                                RKGS  60
      GOTO 10                                                           RKGS  61
C     END OF INNERMOST RUNGE-KUTTA LOOP                                 RKGS  62
C     TEST OF ACCURACY                                                  RKGS  63
   15 IF(ITEST)16,16,20                                                 RKGS  64
C     IN CASE ITEST=0 THERE IS NO POSSIBILITY FOR TESTING OF ACCURACY   RKGS  65
   16 DO 17 I=1,NDIM                                                    RKGS  66
   17 AUX(4,I)=Y(I)                                                     RKGS  67
      ITEST=1                                                           RKGS  68
      ISTEP=ISTEP+ISTEP-2                                               RKGS  69
   18 IHLF=IHLF+1                                                       RKGS  70
      X=X-H                                                             RKGS  71
      H=.5*H                                                            RKGS  72
      DO 19 I=1,NDIM                                                    RKGS  73
      Y(I)=AUX(1,I)                                                     RKGS  74
      DERY(I)=AUX(2,I)                                                  RKGS  75
   19 AUX(6,I)=AUX(3,I)                                                 RKGS  76
      GOTO 9                                                            RKGS  77
C     IN CASE ITEST=1 TESTING OF ACCURACY IS POSSIBLE                   RKGS  78
   20 IMOD=ISTEP/2                                                      RKGS  79
      IF(ISTEP-IMOD-IMOD)21,23,21                                       RKGS  80
   21 CALL FCT(X,Y,DERY)                                                RKGS  81
      DO 22 I=1,NDIM                                                    RKGS  82
      AUX(5,I)=Y(I)                                                     RKGS  83
   22 AUX(7,I)=DERY(I)                                                  RKGS  84
      GOTO 9                                                            RKGS  85
C     COMPUTATION OF TEST VALUE DELT                                    RKGS  86
   23 DELT=0.                                                           RKGS  87
      DO 24 I=1,NDIM                                                    RKGS  88
   24 DELT=DELT+AUX(8,I)*ABS(AUX(4,I)-Y(I))                             RKGS  89
      IF(DELT-PRMT(4))28,28,25                                          RKGS  90
C     ERROR IS TOO GREAT                                                RKGS  91
   25 IF(IHLF-10)26,36,36                                               RKGS  92
   26 DO 27 I=1,NDIM                                                    RKGS  93
   27 AUX(4,I)=AUX(5,I)                                                 RKGS  94
      ISTEP=ISTEP+ISTEP-4                                               RKGS  95
      X=X-H                                                             RKGS  96
      IEND=0                                                            RKGS  97
      GOTO 18                                                           RKGS  98
C     RESULT VALUES ARE GOOD                                            RKGS  99
   28 CALL FCT(X,Y,DERY)                                                RKGS 100
      DO 29 I=1,NDIM                                                    RKGS 101
      AUX(1,I)=Y(I)                                                     RKGS 102
      AUX(2,I)=DERY(I)                                                  RKGS 103
      AUX(3,I)=AUX(6,I)                                                 RKGS 104
      Y(I)=AUX(5,I)                                                     RKGS 105
   29 DERY(I)=AUX(7,I)                                                  RKGS 106
      CALL OUTP(X-H,Y,DERY,IHLF,NDIM,PRMT)                              RKGS 107
      IF(PRMT(5))40,30,40                                               RKGS 108
   30 DO 31 I=1,NDIM                                                    RKGS 109
      Y(I)=AUX(1,I)                                                     RKGS 110
   31 DERY(I)=AUX(2,I)                                                  RKGS 111
      IREC=IHLF                                                         RKGS 112
      IF(IEND)32,32,39                                                  RKGS 113
C     INCREMENT GETS DOUBLED                                            RKGS 114
   32 IHLF=IHLF-1                                                       RKGS 115
      ISTEP=ISTEP/2                                                     RKGS 116
      H=H+H                                                             RKGS 117
      IF(IHLF)4,33,33                                                   RKGS 118
   33 IMOD=ISTEP/2                                                      RKGS 119
      IF(ISTEP-IMOD-IMOD)4,34,4                                         RKGS 120
   34 IF(DELT-.02*PRMT(4))35,35,4                                       RKGS 121
   35 IHLF=IHLF-1                                                       RKGS 122
      ISTEP=ISTEP/2                                                     RKGS 123
      H=H+H                                                             RKGS 124
      GOTO 4                                                            RKGS 125
C     RETURNS TO CALLING PROGRAM                                        RKGS 126
   36 IHLF=11                                                           RKGS 127
      CALL FCT(X,Y,DERY)                                                RKGS 128
      GOTO 39                                                           RKGS 129
   37 IHLF=12                                                           RKGS 130
      GOTO 39                                                           RKGS 131
   38 IHLF=13                                                           RKGS 132
   39 CALL OUTP(X,Y,DERY,IHLF,NDIM,PRMT)                                RKGS 133
   40 RETURN                                                            RKGS 134
      END                                                               RKGS 135
```

Mathematics - Fourier Analysis

FORIF

This subroutine produces the Fourier coefficients for a given periodic function.

Given:
1. A function f(x) for values of x between 0 and 2π
2. N - the spacing desired such that the interval is $2\pi/(2N+1)$
3. M - the desired order of the Fourier coefficients, $0 \leq M \leq N$.

The coefficients of the Fourier series that approximate the given function are calculated as follows:

$$C_1 = \cos\left(\frac{2\pi}{2N+1}\right) \tag{1}$$

$$S_1 = \sin\left(\frac{2\pi}{2N+1}\right) \tag{2}$$

$$U_2 = 0$$

$$U_1 = 0$$

$$C = 1$$

$$S = 0$$

$$J = 1$$

The following recursive sequence is used to compute U_0, U_1, and U_2:

$$U_0 = f\left(\frac{2m\pi}{2N+1}\right) + 2\,C\,U_1 - U_2 \tag{3}$$

$$U_2 = U_1$$

$$U_1 = U_0$$

for values of m = 2N, 2N-1, ..., 1

The coefficients are then:

$$A_J = \frac{2}{2N+1}\left(f(0) + C\,U_1 - U_2\right) \tag{4}$$

$$B_J = \frac{2}{2N+1}\,S\,U_1 \tag{5}$$

The values of C and S are updated to:

$$Q = C_1\,C - S_1\,S$$

$$S = C_1\,S + S_1\,C$$

$$C = Q$$

J is stepped by 1 and the sequence starting at equation (3) is now repeated until M+1 pairs of coefficients have been computed.

Subroutine FORIF

Purpose:

Fourier analysis of a given periodic function in the range $0-2\pi$.
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.

Usage:

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

Description of parameters:

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 $(0, 2\pi)$. The spacing is thus $2\pi/(2N+1)$.

M - The maximum order of the harmonics to be fitted.

A - Resultant vector of Fourier cosine coefficients of length M+1; i.e., $A_0, \ldots, A_M$.

B - Resultant vector of Fourier sine coefficients of length M+1; i.e., $B_0, \ldots, B_M$.

IER - Resultant error code where:

IER=0	No error.
IER=1	N not greater than or equal to M.
IER=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 - Name of user function subprogram used for computing data points.

Calling program must have FORTRAN EXTERNAL statement containing names of function subprograms listed in call to FORIF.

Method:

Uses recursive technique described in A. Ralston, H. Wilf, 'Mathematical Methods for Digital Computers', John Wiley and Sons, New York, 1960, Chapter 24. The method of indexing through the procedure has been modified to simplify the computation.

```
      SUBROUTINE FORIF(FUN,N,M,A,B,IER)                                 FORIF  1
      DIMENSION A(1),B(1)                                               FORIF  2
C        CHECK FOR PARAMETER ERRORS                                     FORIF  3
      IER=0                                                             FORIF  4
   20 IF(M) 30,40,40                                                    FORIF  5
   30 IER=2                                                             FORIF  6
      RETURN                                                            FORIF  7
   40 IF(M-N) 60,60,50                                                  FORIF  8
   50 IER=1                                                             FORIF  9
      RETURN                                                            FORIF 10
C        COMPUTE AND PRESET CONSTANTS                                   FORIF 11
   60 AN=N                                                              FORIF 12
      COEF=2.0/(2.0*AN+1.0)                                             FORIF 13
      CONST=3.141593*COEF                                               FORIF 14
      S1=SIN(CONST)                                                     FORIF 15
      C1=COS(CONST)                                                     FORIF 16
      C=1.0                                                             FORIF 17
      S=0.0                                                             FORIF 18
      J=1                                                               FORIF 19
      FUNZ=FUN(0.0)                                                     FORIF 20
   70 U2=0.0                                                            FORIF 21
      U1=0.0                                                            FORIF 22
      AI=2*N                                                            FORIF 23
C        FORM FOURIER COEFFICIENTS RECURSIVELY                          FORIF 24
   75 X=AI*CONST                                                        FORIF 25
      U0=FUN(X)+2.0*C*U1-U2                                             FORIF 26
      U2=U1                                                             FORIF 27
      U1=U0                                                             FORIF 28
      AI=AI-1.0                                                         FORIF 29
      IF(AI) 80,80,75                                                   FORIF 30
   80 A(J)=COEF*(FUNZ+C*U1-U2)                                          FORIF 31
      B(J)=COEF*S*U1                                                    FORIF 32
      IF(J-(N+1)) 90,130,100                                            FORIF 33
   90 Q=C1*C-S1*S                                                       FORIF 34
      S=C1*S+S1*C                                                       FORIF 35
      C=Q                                                               FORIF 36
      J=J+1                                                             FORIF 37
      GO TO 70                                                          FORIF 38
  100 A(1)=A(1)*0.5                                                     FORIF 39
      RETURN                                                            FORIF 40
      END                                                               FORIF 41
```

FORIT

This subroutine produces the Fourier coefficients of a tabulated function.

Given:

1. Tabulated values of a function f(x) for x between 0 and 2π in steps of $2\pi/(2N+1)$
2. N such that there are 2N+1 tabulated data points: $2K\pi/2N+1$, K = 0, 1, 2, ..., 2N
3. M - the desired order of the Fourier coefficients where $0 \leq M \leq N$

The coefficients of the Fourier series which approximate the given function are calculated as follows:

$$C_1 = \cos\left(\frac{2\pi}{2N+1}\right) \quad (1)$$

$$S_1 = \sin\left(\frac{2\pi}{2N+1}\right) \quad (2)$$

$$U_2 = 0$$

$$U_1 = 0$$

$$C = 1$$

$$S = 0$$

$$J = 1$$

The following recursive sequence is used to compute U_0, U_1, and U_2:

$$U_0 = f\left(\frac{2m\pi}{2N+1}\right) + 2\,C\,U_1 - U_2 \quad (3)$$

$$U_2 = U_1$$

$$U_1 = U_0$$

for values of m = 2N, 2N-1, ..., 1

The coefficients are then:

$$A_J = \frac{2}{2N+1}\left(f(0) + C\,U_1 - U_2\right) \quad (4)$$

$$B_J = \frac{2}{2N+1} S U_1 \qquad (5)$$

The values of C and S are updated to:

$$Q = C_1 C - S_1 S$$

$$S = C_1 S + S_1 C$$

$$C = Q$$

J is stepped by 1 and the sequence starting at equation (3) is now repeated until M+1 pairs of coefficients have been computed.

Subroutine FORIT

Purpose:

Fourier analysis of a periodically tabulated function.

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 a given set of periodically tabulated values of a function.

Usage:

CALL FORIT(FNT, N, M, A, B, IER)

Description of parameters:

- FNT - Vector of tabulated function values of length 2N+1.
- N - Defines the interval such that 2N+1 points are taken over the interval $(0, 2\pi)$. The spacing is thus $2\pi/(2N+1)$.
- M - Maximum order of harmonics to be fitted.
- A - Resultant vector of Fourier cosine coefficients of length M+1; i.e., $A_0, \ldots, A_M$.
- B - Resultant vector of Fourier sine coefficients of length M+1; i.e., $B_0, \ldots, B_M$.
- IER - Resultant error code where:
  - IER=0 No error.
  - IER=1 N not greater or equal to M.
  - IER=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 of vector B is zero in all cases.

Subroutines and function subprograms required:

None.

Method:

Uses recursive technique described in A. Ralston, H. Wilf, 'Mathematical Methods for Digital Computers', John Wiley and Sons, New York, 1960, Chapter 24. The method of indexing through the procedure has been modified to simplify the computation.

```
      SUBROUTINE FORIT(FNT,N,M,A,B,IER)                                 FORIT  1
      DIMENSION A(1),B(1),FNT(1)                                        FORIT  2
C        CHECK FOR PARAMETER ERRORS                                     FORIT  3
      IER=0                                                             FORIT  4
   20 IF(M) 30,40,40                                                    FORIT  5
   30 IER=2                                                             FORIT  6
      RETURN                                                            FORIT  7
   40 IF(M-N) 60,60,50                                                  FORIT  8
   50 IER=1                                                             FORIT  9
      RETURN                                                            FORIT 10
C        COMPUTE AND PRESET CONSTANTS                                   FORIT 11
   60 AN=N                                                              FORIT 12
      COEF=2.0/(2.0*AN+1.0)                                             FORIT 13
      CONST=3.141593*COEF                                               FORIT 14
      S1=SIN(CONST)                                                     FORIT 15
      C1=COS(CONST)                                                     FORIT 16
      C=1.0                                                             FORIT 17
      S=0.0                                                             FORIT 18
      J=1                                                               FORIT 19
      FNTZ=FNT(1)                                                       FORIT 20
   70 U2=0.0                                                            FORIT 21
      U1=0.0                                                            FORIT 22
      I=2*N+1                                                           FORIT 23
C        FORM FOURIER COEFFICIENTS RECURSIVELY                          FORIT 24
   75 U0=FNT(I)+2.0*C*U1-U2                                             FORIT 25
      U2=U1                                                             FORIT 26
      U1=U0                                                             FORIT 27
      I=I-1                                                             FORIT 28
      IF(I-1) 80,80,75                                                  FORIT 29
   80 A(J)=COEF*(FNTZ+C*U1-U2)                                          FORIT 30
      B(J)=COEF*S*U1                                                    FORIT 31
      IF(J-(M+1)) 90,100,100                                            FORIT 32
   90 Q=C1*C-S1*S                                                       FORIT 33
      S=C1*S+S1*C                                                       FORIT 34
      C=Q                                                               FORIT 35
      J=J+1                                                             FORIT 36
      GO TO 70                                                          FORIT 37
  100 A(1)=A(1)*0.5                                                     FORIT 38
      RETURN                                                            FORIT 39
      END                                                               FORIT 40
```

Mathematics - Special Operations and Functions

GAMMA

This subroutine computes the value of the gamma function for a given argument x.

Calculation of the Gamma Function. $\Gamma(x)$ is defined for $x > 0$ by:

$$\Gamma(x) = \int_0^\infty t^{x-1} \cdot e^{-t}\, dt \qquad (1)$$

This function satisfies the recurrence relation:

$$\Gamma(x) = (x-1) \cdot \Gamma(x-1) \qquad (2)$$

which defines $\Gamma(x)$ for any x not a negative integer.

Note that when x is a positive integer $\Gamma(x) = (x-1)!$

To compute $\Gamma(x)$ for $x > 1$, apply the recurrence (2), r times until $1 < x - r = y \leq 2$. Thus, for $x > 1$

$$\Gamma(x) = (x-1)(x-2) \ldots (x-r)\, \Gamma(y) \qquad (3)$$

$\Gamma(y)$ is computed from the following formula:

$$\Gamma(y) \approx 1 - 0.57710166(y-1) + 0.98585399(y-1)^2$$

$$- 0.87642182(y-1)^3 + 0.83282120(y-1)^4$$

$$- 0.56847290(y-1)^5 + 0.25482049(y-1)^6$$

$$- 0.05149930(y-1)^7 \qquad (4)$$

For $x < 1$, the recurrence (2) is taken in the direction of decreasing n, giving

$$\Gamma(x) = \frac{\Gamma(y)}{x(x+1)(x+2)\ldots(x+r-1)} \qquad (5)$$

where $1 < x + r = y \leq 2$.

As before, $\Gamma(y)$ is computed using equation (4).

Subroutine GAMMA

Purpose:
Computes the gamma function for a given argument.

Usage:
CALL GAMMA(XX, GX, IER)

Description of parameters:
XX - The argument for the gamma function.
GX - The resultant gamma function value.
IER - Resultant error code where:
IER=0 No error.
IER=1 XX is within .000001 of being a negative integer.
IER=2 XX is greater than 34.5 GX is set to 1.0E38

Remarks:
None.

Subroutines and function subprograms required:
None.

Method:
The recursion relation and polynomial approximation by C. Hastings, Jr., 'Approximations for Digital Computers', Princeton University Press, 1955.

```
      SUBROUTINE GAMMA(XX,GX,IER)                                       GAMMA  1
      IF(XX-34.5)16,6,4                                                 GAMMAM01
    4 IER=2                                                             GAMMAM02
      GX=1.E38                                                          GAMMAM03
      RETURN                                                            GAMMAM04
    6 X=XX                                                              GAMMAM05
      ERR=1.0E-6                                                        GAMMA  3
      IER=0                                                             GAMMA  4
      GX=1.0                                                            GAMMA  5
      IF(X-2.0)50,50,15                                                 GAMMA  6
   10 IF(X-2.0)110,110,15                                               GAMMA  7
   15 X=X-1.0                                                           GAMMA  8
      GX=GX*X                                                           GAMMA  9
      GO TO 10                                                          GAMMA 10
   50 IF(X-1.0)60,120,110                                               GAMMA 11
C        SEE IF X IS NEAR NEGATIVE INTEGER OR ZERO                      GAMMA 12
   60 IF(X-ERR)62,62,80                                                 GAMMA 13
   62 K=X                                                               GAMMA 14
      Y=FLOAT(K)-X                                                      GAMMA 15
      IF(ABS(Y)-ERR)130,130,64                                          GAMMA 16
   64 IF(1.0-Y-ERR)130,130,70                                           GAMMA 17
C        X NOT NEAR A NEGATIVE INTEGER OR ZERO                          GAMMA 18
   70 IF(X-1.0)80,80,110                                                GAMMA 19
   80 GX=GX/X                                                           GAMMA 20
      X=X+1.0                                                           GAMMA 21
      GO TO 70                                                          GAMMA 22
  110 Y=X-1.0                                                           GAMMA 23
      GY=1.0+Y*(-0.5771017+Y*(+0.9858540+Y*(-0.8764218+Y*(+0.8328212+   GAMMA 24
     1Y*(-0.5684729+Y*(+0.2548205+Y*(-0.05149930)))))))                 GAMMA 25
      GX=GX*GY                                                          GAMMA 26
  120 RETURN                                                            GAMMA 27
  130 IER=1                                                             GAMMA 28
      RETURN                                                            GAMMA 29
      END                                                               GAMMA 30
```

LEP

This subroutine computes the values of the Legendre polynomials for a given argument x and orders zero up to N. The Legendre polynomial $P_n(x)$ satisfies the recurrence equation

$$P_{n+1}(x) = ((2n+1) \cdot x \cdot P_n(x) - n \cdot P_{n-1}(x))/(n+1)$$

with starting values $P_0(x) = 1$, $P_1(x) = x$.

For reasons of economy and numerical stability the recurrence equation is used in the form:

$$P_{n+1}(x) = x \cdot P_n(x) - P_{n-1}(x) + x \cdot P_n(x) - (x \cdot P_n(x) - P_{n-1}(x))/(n+1)$$

For large values of n the last term is negligible, giving the approximation:

$$P_{n+1}(x) = 2 \cdot x \cdot P_n(x) - P_{n-1}(x)$$

This form shows that roundoff errors grow at worst linearly, assuming that the argument x is absolutely less than one.

If e_{n+r} is the error in $P_{n+r}(x)$ due to a single rounding error e in $P_n(x)$, the approximation is

$$e_{n+r+1} = 2x \cdot e_{n+r} - e_{n+r-1}$$

with initial conditions $e_n = e$, $e_{n-1} = 0$. The solution of this difference equation has its maximum for $|x| = 1$:

$$e_{n-1} = 0,\ e_n = e,\ |e_{n+1}| = 2e,\ \ldots,\ |e_{n+r}| = (r+1)e$$

The order is assumed to be zero for negative values of N.

Subroutine LEP

Purpose:
Compute the values of the Legendre polynomials P(N,X) for argument value X and orders 0 to N.

Usage:
CALL LEP(Y,X,N)

Description of parameters:
Y - Result vector of dimension N+1 containing the values of Legendre polynomials of order 0 to N for given argument X. Values are ordered from low to high order.
X - Argument of Legendre polynomial.
N - Order of Legendre polynomial.

Remarks:

N less than 0 is treated as if N were 0.

Subroutines and function subprograms required:

None.

Method:

Evaluation is based on the recurrence equation for Legendre polynomials P(N,X); P(N+1,X)=2*X*P(N,X)-P(N-1,X)-(X*P(N,X)-P(N-1,X))/(N+1), where the first term in brackets is the order, and the second is the argument.

Starting values are P(0,X)=1, P(1,X)=X.

```
      SUBROUTINE LEP(Y,X,N)                                          LEP    1
      DIMENSION Y(1)                                                 LEP    2
C        TEST OF ORDER                                               LEP    3
      L1=1                                                           LEP  M01
      L2=2                                                           LEP  M02
      Y(L1)=1.0                                                      LEP  M03
      IF(N)1,1,2                                                     LEP    5
    1 RETURN                                                         LEP    6
    2 Y(L2)=X                                                        LEP  M04
      IF(N-1)1,1,3                                                   LEP    8
    3 DO 4 I=2,N                                                     LEP    9
      G=X*Y(I)                                                       LEP   10
    4 Y(I+1)=G-Y(I-1)  -(G-Y(I-1))/FLOAT(I)+G                        LEP   11
      RETURN                                                         LEP   12
      END                                                            LEP   13
```

BESJ

This subroutine computes the J Bessel function for a given argument and integer order by using the recurrence relationship:

$$F_{n+1}(x) + F_{n-1}(x) = \left(\frac{2n}{x}\right) F_n(x) \qquad (1)$$

The desired Bessel function is:

$$J_n(x) = \frac{F_n(x)}{\alpha} \qquad (2)$$

where

$$\alpha = F_0(x) + 2 \sum_{m=1}^{M-2} F_{2m}(x) \qquad (3)$$

M is initialized at M_0.

M_0 is the greater of M_A and M_B where:

$M_A = [x+6]$ if $x < 5$ and $M_A = [1.4x+60/x]$ if $x \geq 5$.

$M_B = [n+x/4+2]$

$F_{M-2}, F_{M-3}, \ldots, F_2, F_1, F_0$ is evaluated using equation (1) with $F_M = 0$ and $F_{M-1} = 10^{-30}$.

α and $J_n(x)$ are then computed using equations (3) and (2) respectively.

The computation is repeated for M+3.

The values of $J_n(x)$ for M and M+3 are compared:

$$\text{If } \left| J_n(x)_M - J_n(x)_{M+3} \right| \leq \delta \left| J_n(x)_{M+3} \right|$$

this value is accepted as $J_n(x)$; if not, the computation is repeated by adding 3 to M and using this as a new value for M. If M reaches M_{MAX} before the desired accuracy is obtained, execution is terminated. M_{MAX} is defined as:

$$M_{MAX} = \begin{cases} \left[20 + 10x - \frac{x^2}{3}\right] & \text{for } x \leq 15 \\ [90 + x/2] & \text{for } x > 15 \end{cases} \qquad (4)$$

Subroutine BESJ

Purpose:

Compute the J Bessel function for a given argument and order.

Usage:

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

Description of parameters:

- 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 where:
  - IER=0 No error.
  - IER=1 N is negative.
  - IER=2 X is negative or zero.
  - IER=3 Required accuracy not obtained.
  - IER=4 Range of N compared to X not correct. (See Remarks.)

Remarks:

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.

Subroutines and function subprograms required:

None.

Method:

Recurrence relation technique described by H. Goldstein and R.M. Thaler, 'Recurrence Techniques for the Calculation of Bessel Functions', M.T.A.C., V.13, pp.102-108 and I.A. Stegun and M. Abramowitz, 'Generation of Bessel Functions on High Speed Computers', M.T.A.C., V.11, 1957, pp.255-257.

```
      SUBROUTINE BESJ(X,N,BJ,D,IER)                                     BESJ   1
      BJ=.0                                                             BESJ   2
      IF(N)10,20,20                                                     BESJ   3
   10 IER=1                                                             BESJ   4
      RETURN                                                            BESJ   5
   20 IF(X)30,30,31                                                     BESJ   6
   30 IER=2                                                             BESJ   7
      RETURN                                                            BESJ   8
   31 IF(X-15.)32,32,34                                                 BESJ   9
   32 NTEST=20.+10.*X-X** 2/3                                           BESJ  10
      GO TO 36                                                          BESJ  11
   34 NTEST=90.+X/2.                                                    BESJ  12
   36 IF(N-NTEST)40,38,38                                               BESJ  13
   38 IER=4                                                             BESJ  14
      RETURN                                                            BESJ  15
   40 IER=0                                                             BESJ  16
      N1=N+1                                                            BESJ  17
      BPREV=.0                                                          BESJ  18
C        COMPUTE STARTING VALUE OF M                                     BESJ  19
      IF(X-5.)50,60,60                                                  BESJ  20
   50 MA=X+6.                                                           BESJ  21
      GO TO 70                                                          BESJ  22
   60 MA=1.4*X+60./X                                                    BESJ  23
   70 MB=N+IFIX(X)/4+2                                                  BESJ  24
      MZERO=MA                                                          BESJ  25
      IF(MA-MB)80,90,90                                                 BESJ  26
   80 MZERO=MB                                                          BESJ  27
C        SET UPPER LIMIT OF M                                            BESJ  28
   90 MMAX=NTEST                                                        BESJ  29
  100 DO 190 M=MZERO,MMAX,3                                             BESJ  30
C        SET F(M1),F(M-1)                                                BESJ  31
      FM1=1.0E-28                                                       BESJ  32
      FM=.0                                                             BESJ  33
      ALPHA=.0                                                          BESJ  34
      IF(M-(M/2)*2)120,110,120                                          BESJ  35
  110 JT=-1                                                             BESJ  36
      GO TO 130                                                         BESJ  37
  120 JT=1                                                              BESJ  38
  130 M2=M-2                                                            BESJ  39
      DO 160 K=1,M2                                                     BESJ  40
      MK=M-K                                                            BESJ  41
      BMK=2.*FLOAT(MK)*FM1/X-FM                                         BESJ  42
      FM=FM1                                                            BESJ  43
      FM1=BMK                                                           BESJ  44
      IF(MK-N-1)150,140,150                                             BESJ  45
  140 BJ=BMK                                                            BESJ  46
  150 JT=-JT                                                            BESJ  47
      S=1+JT                                                            BESJ  48
  160 ALPHA=ALPHA+BMK*S                                                 BESJ  49
      BMK=2.*FM1/X-FM                                                   BESJ  50
      IF(N)180,170,180                                                  BESJ  51
  170 BJ=BMK                                                            BESJ  52
  180 ALPHA=ALPHA+BMK                                                   BESJ  53
      BJ=BJ/ALPHA                                                       BESJ  54
      IF(ABS(BJ-BPREV)-ABS(D*BJ))200,200,190                            BESJ  55
  190 BPREV=BJ                                                          BESJ  56
      IER=3                                                             BESJ  57
  200 RETURN                                                            BESJ  58
      END                                                               BESJ  59
```

BESY

This subroutine computes the Y Bessel function for a given argument x and order n. The recurrence relation:

$$Y_{n+1}(x) = \left(\frac{2n}{x}\right) \cdot Y_n(x) - Y_{n-1}(x) \tag{1}$$

is used for this evaluation.

For $x > 4$

$$Y_0(x) = \sqrt{\frac{2}{\pi x}} \left(P_0(x) \sin\left(x - \frac{\pi}{4}\right) + Q_0(x) \cos\left(x - \frac{\pi}{4}\right) \right) \tag{2}$$

$$Y_1(x) = \sqrt{\frac{2}{\pi x}} \left(-P_1(x) \cos\left(x - \frac{\pi}{4}\right) + Q_1(x) \sin\left(x - \frac{\pi}{4}\right) \right) \tag{3}$$

$P_0(x)$, $Q_0(x)$, $P_1(x)$, and $Q_1(x)$ are:

$$\frac{1}{\sqrt{2\pi}} P_0\left(\frac{4}{t}\right) = 0.3989422793 - 0.0017530620t^2 + 0.0001734300t^4 - 0.0000487613t^6 + 0.0000173565t^8 - 0.0000037043t^{10} \tag{4}$$

$$\frac{1}{t\sqrt{2\pi}} Q_0\left(\frac{4}{t}\right) = -0.124669441 + 0.0004564324t^2 - 0.0000869791t^4 + 0.0000342468t^6 - 0.0000142078t^8 + 0.0000032312t^{10} \tag{5}$$

$$\frac{1}{\sqrt{2\pi}} P_1\left(\frac{4}{t}\right) = 0.3989422819 + 0.0029218256t^2 - 0.0002232030t^4 + 0.0000580759t^6 - 0.0000200920t^8 + 0.0000042414t^{10} \tag{6}$$

$$\frac{1}{t\sqrt{2\pi}} Q_1\left(\frac{4}{t}\right) = 0.0374008364 - 0.0006390400t^2 + 0.0001064741t^4 - 0.0000398708t^6 + 0.0000162200t^8 - 0.0000036594t^{10} \tag{7}$$

where $t = \frac{4}{x}$

For $x \leq 4$

$$Y_0(x) = \frac{2}{\pi} \sum_{m=0}^{15} (-1)^m \left(\frac{x}{2}\right)^{2m} \frac{1}{(m!)^2} \left[\log\frac{x}{2} + \gamma - H_m\right] \tag{8}$$

where

$$H_m = \sum_{r=1}^{m} \frac{1}{r} \text{ if } m \geq 1 = 0 \quad \text{if } m = 0 \tag{9}$$

and γ = Euler's constant = 0.5772156649

$$Y_1(x) = -\frac{2}{\pi x} + \frac{2}{\pi} \sum_{m=1}^{16} (-1)^{m+1} \left(\frac{x}{2}\right)^{2m-1} \frac{1}{m!\,(m-1)!} \cdot \left[\log\frac{x}{2} + \gamma - H_m + \frac{1}{2m}\right] \tag{10}$$

Subroutine BESY

Purpose:
Compute the Y Bessel function for a given argument and order.

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

Description of parameters:
- 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 where:

IER= 0	No error.
IER= 1	N is negative.
IER= 2	X is negative or zero.
IER= 3	BY is greater than 10**36.

Remarks:

Very small values of X may cause the range of the library function ALOG to be exceeded. For $N > 30$ and $X \leq 5$, this condition may occur.
X must be greater than zero.
N must be greater than or equal to zero.

Subroutines and function subprograms required:

None.

Method:

Recurrence relation and polynomial approximation technique as described by A.J.M. Hitchcock, 'Polynomial Approximations to Bessel Functions of Order Zero and One and to Related Functions', M.T.A.C., V.11, 1957, pp.86-88, and G.N. Watson, 'A Treatise on the Theory of Bessel Functions', Cambridge University Press, 1958 p. 62.

```
      SUBROUTINE BESY(X,N,BY,IER)                                       BESY   1
C       CHECK FOR ERRORS IN N AND X                                     BESY   2
      IF(N)180,10,10                                                    BESY   3
   10 IER=0                                                             BESY   4
      IF(X)190,190,20                                                   BESY   5
C       BRANCH IF X LESS THAN OR EQUAL 4                                BESY   7
   20 IF(X-4.0)40,40,30                                                 BESY M05
C       COMPUTE Y0 AND Y1 FOR X GREATER THAN 4                          BESY   9
   30 T1=4.0/X                                                          BESY M06
      T2=T1*T1                                                          BESY M07
      P0=((((-.00000037043*T2+.0000173565)*T2-.0000487613)*T2           BESY M08
     1  +.00017343)*T2-.001753062)*T2+.3989423                          BESY M09
      Q0=((((.0000032312*T2-.0000142078)*T2+.0000342468)*T2             BESY M10
     1  -.0000869791)*T2+.0004564324)*T2-.01246694                      BESY M11
      P1=((((.00000424 14*T2-.0000200920)*T2+.0000580759)*T2            BESY M12
     1  -.000223203)*T2+.002921826)*T2+.3989423                         BESY M13
      Q1=((((-.0000036594*T2+.000016221)*T2-.000039870 8)*T2            BESY M14
     1  +.0001064741)*T2-.0006390400)*T2+.03740084                      BESY M15
      A=2.0/SQRT(X)                                                     BESY M16
      B=A*T1                                                            BESY M17
      C=X-.78539 82                                                     BESY M18
      Y0=A*P0*SIN(C)+B*Q0*COS(C)                                        BESY M19
      Y1=-A*P1*COS(C)+B*Q1*SIN(C)                                       BESY M20
      GO TO 90                                                          BESY  51
C       COMPUTE Y0 AND Y1 FOR X LESS THAN OR EQUAL TO 4                 BESY  52
   40 XX=X/2.                                                           BESY  53
      X2=XX*XX                                                          BESY  54
      T=ALOG(XX)+.5772157                                               BESY M21
      SUM=0.                                                            BESY  56
      TERM=T                                                            BESY  57
      Y0=T                                                              BESY  58
      DO 70 L=1,15                                                      BESY  59
      IF(L-1)50,60,50                                                   BESY  60
   50 SUM=SUM+1./FLOAT(L-1)                                             BESY  61
   60 FL=L                                                              BESY  62
      TS=T-SUM                                                          BESY  63
      TERM=(TERM*(-X2)/FL**2)*(1.-1./(FL*TS))                           BESY  64
   70 Y0=Y0+TERM                                                        BESY  65
      TERM = XX*(T-.5)                                                  BESY  66
      SUM=0.                                                            BESY  67
      Y1=TERM                                                           BESY  68
      DO 80 L=2,16                                                      BESY  69
      SUM=SUM+1./FLOAT(L-1)                                             BESY  70
      FL=L                                                              BESY  71
      FL1=FL-1.                                                         BESY  72
      TS=T-SUM                                                          BESY  73
      TERM=(TERM*(-X2)/(FL1*FL))*((TS-.5/FL)/(TS+.5/FL1))               BESY  74
   80 Y1=Y1+TERM                                                        BESY  75
      PI2=.6366198                                                      BESY M22
      Y0=PI2*Y0                                                         BESY  77
      Y1=-PI2/X+PI2*Y1                                                  BESY  78
C       CHECK IF ONLY Y0 OR Y1 IS DESIRED                               BESY  79
   90 IF(N-1)100,100,130                                                BESY  80
C       RETURN EITHER Y0 OR Y1 AS REQUIRED                              BESY  81
  100 IF(N)110,120,110                                                  BESY  82
  110 BY=Y1                                                             BESY  83
      GO TO 170                                                         BESY  84
  120 BY=Y0                                                             BESY  85
      GO TO 170                                                         BESY  86
C       PERFORM RECURRENCE OPERATIONS TO FIND YN(X)                     BESY  87
  130 YA=Y0                                                             BESY  88
      YB=Y1                                                             BESY  89
      K=1                                                               BESY  90
  140 T=FLOAT(2*K)/X                                                    BESY  91
      YC=T*YB-YA                                                        BESY  92
      IF(ABS(YC)-1.0E36)145,145,141                                     BESY M01
  141 IER=3                                                             BESY M02
      RETURN                                                            BESY M03
  145 K=K+1                                                             BESY M04
      IF(K-N)150,160,160                                                BESY  94
  150 YA=YB                                                             BESY  95
      YB=YC                                                             BESY  96
      GO TO 140                                                         BESY  97
  160 BY=YC                                                             BESY  98
  170 RETURN                                                            BESY  99
  180 IER=1                                                             BESY 100
      RETURN                                                            BESY 101
  190 IER=2                                                             BESY 102
      RETURN                                                            BESY 103
      END                                                               BESY 104
```

BESI

This subroutine computes the I Bessel function for a given argument x and order n.

For $x \leq 12$ or $\leq n$

$$I_n(x) = \left(\frac{x}{2}\right)^n \frac{1}{n!} \sum_{s=0}^{30} \left(\frac{x}{2}\right)^{2s} \frac{n!}{s!(n+s)!} \qquad (1)$$

For $x > 12$ and $> n$

$$I_n(x) = \frac{e^x}{\sqrt{2\pi x}} \sum_{m=0}^{30} (8x)^{-m} \cdot \frac{1}{m!}$$

$$\prod_{K=1}^{m} \left((2K-1)^2 - 4n^2\right) \qquad (2)$$

Subroutine BESI

Purpose:

Compute the I Bessel function for a given argument and order.

Usage:

CALL BESI(X, N, BI, IER)

Description of parameters:

- 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 where:
  - IER= 0 No error.
  - IER= 1 N is negative.
  - IER= 2 X is negative.
  - IER= 3 BI is less than 1.0E-36, and is set to zero.
  - IER= 4 X is greater than 60 and and greater than N.

Remarks:

X and N must be greater than zero.

Subroutines and function subprograms required:

None.

Method:

Computes the I^{th} Bessel function using series or asymptotic approximations depending on the range of the arguments.

```
      SUBROUTINE BESI(X,N, BI,IER)                                      BESI   1
C        CHECK FOR ERRORS IN N AND X AND EXIT IF ANY ARE PRESENT         BESI   2
      IER=0                                                             BESI   3
      BI=1.0                                                            BESI   4
      IF(N)150,15,10                                                    BESI   5
   10 IF(X)160,20,20                                                    BESI   6
   15 IF(X)160,17,20                                                    BESI   7
   17 RETURN                                                            BESI   8
C        DEFINE TOLERANCE                                               BESI   9
   20 TOL=1.E-6                                                         BESI  10
C        IF ARGUMENT GT 12 AND GT N, USE ASYMPTOTIC FORM                BESI  11
      IF(X-12.)40,40,30                                                 BESI  12
   30 IF(X-FLOAT(N))40,40,110                                           BESI  13
C        COMPUTE FIRST TERM OF SERIES AND SET INITIAL VALUE OF THE SUM   BESI  14
   40 XX=X/2.                                                           BESI  15
   50 TERM=1.0                                                          BESI M01
      IF(N) 70,70,55                                                    BESI M02
   55 DO 60 I=1,N                                                       BESI M03
      FI=I                                                              BESI M04
      IF(ABS(TERM)-1.E-36)56,60,60                                      BESI M05
   56 IER=3                                                             BESI M06
      BI=0.0                                                            BESI M07
      RETURN                                                            BESI M08
   60 TERM=TERM*XX/FI                                                   BESI M09
   70 BI=TERM                                                           BESI M10
      XX=XX*XX                                                          BESI  23
C        COMPUTE TERMS,STOPPING WHEN ABS(TERM) LE ABS(SUM OF TERMS)     BESI M11
C        TIMES TOLERANCE                                                BESI M12
      DO 90 K=1,1000                                                    BESI M13
      IF(ABS(TERM)-ABS(BI*TOL))100,100,80                               BESI  27
   80 FK=K*(N+K)                                                        BESI  28
      TERM=TERM*(XX/FK)                                                 BESI  29
   90 BI=BI+TERM                                                        BESI  30
C        RETURN BI AS ANSWER                                            BESI  31
  100 RETURN                                                            BESI  32
C        X GT 12 AND X GT N, SO USE ASYMPTOTIC APPROXIMATION            BESI  33
  110 FN=4*N*N                                                          BESI  34
      IF(X- 60.0)115,111,111                                            BESI M14
  111 IER=4                                                             BESI M15
      RETURN                                                            BESI M16
  115 XX=1./(8.*X)                                                      BESI M17
      TERM=1.                                                           BESI  36
      BI=1.                                                             BESI  37
      DO 130 K=1,30                                                     BESI  38
      IF(ABS(TERM)-ABS(TOL*BI))140,140,120                              BESI  39
  120 FK=(2*K-1)**2                                                     BESI  40
      TERM=TERM*XX*(FK-FN)/FLOAT(K)                                     BESI  41
  130 BI=BI+TERM                                                        BESI  42
C        SIGNIFICANCE LOST AFTER 30 TERMS,TRY SERIES                    BESI M18
      GO TO 40                                                          BESI M19
  140 PI=3.141592653                                                    BESI  43
      BI=BI*EXP(X)/SQRT(2.*PI*X)                                        BESI  44
      GO TO 100                                                         BESI  45
  150 IER=1                                                             BESI  46
      GO TO 100                                                         BESI  47
  160 IER=2                                                             BESI  48
      GO TO 100                                                         BESI  49
      END                                                               BESI  50
```

BESK

This subroutine computes the K Bessel function for a given argument x and order n.

The recurrence relation:

$$K_{n+1}(x) = \frac{2n}{x} K_n(x) + K_{n-1}(x) \tag{1}$$

is used for this evaluation.

The initial values K_0 and K_1 are found as follows:

For $x > 1$

$$K_0(x) = e^{-x} \sqrt{\frac{\pi}{2x}}\, G_0(x) \tag{2}$$

$$K_1(x) = e^{-x} \sqrt{\frac{\pi}{2x}}\, G_1(x) \tag{3}$$

where $x = 1/t$ for $t < 1$

$$G_0\left(\frac{1}{t}\right)\cdot\sqrt{\frac{\pi}{2}} = 1.2533141373 - 0.1566641816t + 0.0881112782t^2 - 0.0913909546t^3 + 0.1344596228t^4 - 0.2299850328t^5 + 0.3792409730t^6 - 0.5247277331t^7 + 0.5575368367t^8 - 0.4262632912t^9 + 0.2184518096t^{10} - 0.0668097672t^{11} + 0.0091893830t^{12} \tag{4}$$

$$G_1\left(\frac{1}{t}\right)\cdot\sqrt{\frac{\pi}{2}} = 1.2533141373 + 0.4699927013t - 0.1468582957t^2 + 0.1280426636t^3 - 0.1736431637t^4 + 0.2847618149t^5 - 0.4594342117t^6 + 0.6283380681t^7 - 0.6632295430t^8 + 0.5050238576t^9 - 0.2581303765t^{10} + 0.0788000118t^{11} - 0.0108241775t^{12} \tag{5}$$

For $x \leq 1$

γ = Euler's constant = 0.5772156649 (6)

$$K_0(x) = -\left(\gamma + \log\frac{x}{2}\right) + \sum_{s=1}^{6} \left(\frac{x}{2}\right)^{2s} \frac{1}{(s!)^2} \left[H_s - \left(\gamma + \log\frac{x}{2}\right)\right] \tag{7}$$

where

$$H_s = \sum_{r=1}^{s} \frac{1}{r} \tag{8}$$

$$K_1(x) = \frac{1}{x} + \sum_{s=1}^{8} \left(\frac{x}{2}\right)^{2s-1} \frac{1}{(s!)^2} \left[\frac{1}{2} + s \cdot \left(\gamma + \log\frac{x}{2} - H_s\right)\right] \tag{9}$$

Subroutine BESK

Purpose:

Compute the K Bessel function for a given argument and order.

Usage:

CALL BESK(X, N, BK, IER)

Description of parameters:

- 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 where:
  - IER= 0 No error.
  - IER= 1 N is negative.
  - IER= 2 X is zero or negative.
  - IER= 3 X is greater than 60. Machine range exceeded.
  - IER= 4 BK is greater than 1. E36.

Remarks:

N must be greater than or equal to zero.

Subroutines and function subprograms required:

None.

Method:

Computes zero-order and first-order Bessel functions using series approximations and then computes N^{th} order function using recurrence relation.

Recurrence relation and polynomial approximation technique as described by A. J. M. Hitchcock, 'Polynomial Approximations to Bessel Functions of Order Zero and One and to Related Functions', M.T.A.C., V.11, 1957, pp. 86-88, and G. N. Watson, 'A Treatise on the Theory of Bessel Functions', Cambridge University Press, 1958, p. 62.

```
      SUBROUTINE BESK(X,N,BK,IER)                                       BESK   1
      DIMENSION T(12)                                                   BESK   2
      AK=.0                                                             BESK   3
      IF(N)10,11,11                                                     BESK   4
   10 IER=1                                                             BESK   5
      RETURN                                                            BESK   6
   11 IF(X)12,12,20                                                     BESK   7
   12 IER=2                                                             BESK   8
      RETURN                                                            BESK   9
   20 IF(X- 60.0)22,22,21                                               BESK M01
   21 IER=3                                                             BESK M02
      RETURN                                                            BESK M03
   22 IER=0                                                             BESK M04
      IF(X-1.)36,36,25                                                  BESK  11
   25 A=EXP(-X)                                                         BESK  12
      B=1./X                                                            BESK  13
      C=SQRT(B)                                                         BESK  14
      T(1)=B                                                            BESK  15
      DO 26 L=2,12                                                      BESK  16
   26 T(L)=T(L-1)*B                                                     BESK  17
      IF(N-1)27,29,27                                                   BESK  18
C        COMPUTE KO USING POLYNOMIAL APPROXIMATION                      BESK  19
   27 G0=A*(1.2533141+-.15666418*T(1)+.088111278*T(2)-.091390954*T(3)   BESK  20
     2+.13445962*T(4)-.22998503*T(5)+.37924097*T(6)-.52472773*T(7)      BESK  21
     3+.55753684*T(8)-.42626329*T(9)+.21845181*T(10)-.06680967*T(11)    BESK  22
     4+.009189383*T(12))*C                                              BESK  23
      IF(N)20,28,29                                                     BESK  24
   28 BK=G0                                                             BESK  25
      RETURN                                                            BESK  26
C        COMPUTE K1 USING POLYNOMIAL APPROXIMATION                      BESK  27
   29 G1=A*(1.2533141+.46999270*T(1)-.14685830*T(2)+.12804266*T(3)      BESK  28
     2=.17364316*T(4)+.28476181*T(5)-.45943421*T(6)+.62833807*T(7)      BESK  29
     3-.66322954*T(8)+.50502386*T(9)-.25813038*T(10)+.078800012*T(11)   BESK  30
     4-.010824177*T(12))*C                                              BESK  31
      IF(N-1)20,30,31                                                   BESK  32
   30 BK=G1                                                             BESK  33
      RETURN                                                            BESK  34
C        FROM KO,K1 COMPUTE KN USING RECURRENCE RELATION                BESK  35
   31 DO 35 J=2,N                                                       BESK  36
      GJ=2.*(FLOAT(J)-1.)*G1/X+G0                                       BESK  37
      IF(GJ-1.0E36)33,33,32                                             BESK M05
   32 IER=4                                                             BESK M06
      GO TO 34                                                          BESK M07
   33 G0=G1                                                             BESK M08
   35 G1=GJ                                                             BESK  39
   34 BK=GJ                                                             BESK M09
      RETURN                                                            BESK  41
   36 B=X/2.                                                            BESK  42
      A=.57721566+ALOG(B)                                               BESK  43
      C=B*B                                                             BESK  44
      IF(N-1)37,43,37                                                   BESK  45
C        COMPUTE KO USING SERIES EXPANSION                              BESK  46
   37 G0=-A                                                             BESK  47
      X2J=1.                                                            BESK  48
      FACT=1.                                                           BESK  49
      HJ=.0                                                             BESK  50
      DO 40 J=1,6                                                       BESK  51
      RJ=1./FLOAT(J)                                                    BESK  52
      X2J=X2J*C                                                         BESK  53
      FACT=FACT*RJ*RJ                                                   BESK  54
      HJ=HJ+RJ                                                          BESK  55
   40 G0=G0+X2J*FACT*(HJ-A)                                             BESK  56
      IF(N)43,42,43                                                     BESK  57
   42 BK=G0                                                             BESK  58
      RETURN                                                            BESK  59
C        COMPUTE K1 USING SERIES EXPANSION                              BESK  60
   43 X2J=B                                                             BESK  61
      FACT=1.                                                           BESK  62
      HJ=1.                                                             BESK  63
      G1=1./X+X2J*(.5+A-HJ)                                             BESK  64
      DO 50 J=2,8                                                       BESK  65
      X2J=X2J*C                                                         BESK  66
      RJ=1./FLOAT(J)                                                    BESK  67
      FACT=FACT*RJ*RJ                                                   BESK  68
      HJ=HJ+RJ                                                          BESK  69
   50 G1=G1+X2J*FACT*(.5+(A-HJ)*FLOAT(J))                               BESK  70
      IF(N-1)51,52,51                                                   BESK  71
   52 BK=G1                                                             BESK  72
      RETURN                                                            BESK  73
      END                                                               BESK  74
```

CEL1

This subroutine computes the complete elliptic integral of the first kind. This is defined as:

$$K(k) = \int_0^{\pi/2} \frac{dt}{\sqrt{1-k^2 \sin^2 t}} \cdot \quad 0 \le k < 1$$

An equivalent definition is:

$$K(k) = \int_0^{\infty} \frac{dx}{\sqrt{(1+x^2)\ (1+k_c^2 x^2)}}$$

where k_c is the complementary modulus:

$$k_c^2 + k^2 = 1, \quad 0 < k_c^2 \le 1$$

The subroutine CELI1 calculates K(k) for given modulus k.

The calculation of RES = K(k) is based on the process of the Arithmetic-Geometric Mean.

Starting with the pair of numbers:

$$a_0 = 1, \ g_0 = k_c$$

the sequences of numbers (a_n), (g_n) are generated using the definition:

$$a_n = \frac{1}{2}(a_{n-1} + g_{n-1}), \quad g_n = \sqrt{a_{n-1}\, g_{n-1}}$$

This iterative process is stopped at the N^{th} step, when $a_N = g_N$.

If D is the number of decimal digits in the mantissa of floating-point numbers, then the equality $a_N = g_N$ must be interpreted as $|a_N - g_N|$ is less than $a_N \cdot 10^{-D}$.

Since the sequences (a_n), (g_n) converge quadratically to the same limit (Arithmetico-Geometrical mean) the test for the end of iteration may be replaced by comparing $|a_{N-1} - g_{N-1}|$ against $a_{N-1} \cdot 10^{-D/2}$, thus saving one calculation of the geometrical mean.

The value of $K(k) = \frac{\pi}{2\, a_N}$.

Subroutine CEL1

Purpose:

Calculate complete elliptic integral of first kind.

Usage:

CALL CEL1 (RES, AK, IER)

Description of parameters:

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

Remarks:

For AK=+1,-1 the result is set 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.

Subroutines and function subprograms required:

None.

Method:

Landen's transformation is used for calculation.
Reference:
R. Bulirsch, 'Numerical Calculation of Elliptic Integrals and Elliptic Functions', Handbook Series Special Functions, Numerische Mathematik Vol. 7, 1965, pp. 78-90.

```
      SUBROUTINE CEL1(RES,AK,IER)                                CEL1   1
      IER=0                                                      CEL1   2
C     TEST MODULUS                                               CEL1   3
      GEO=1.-AK*AK                                               CEL1   4
      IF(GEO)1,2,3                                               CEL1   5
    1 IER=1                                                      CEL1   6
      RETURN                                                     CEL1   7
C     SET RESULT VALUE =OFLOW                                    CEL1   8
    2 RES=1.E38                                                  CEL1   9
      RETURN                                                     CEL1  10
    3 GEO=SQRT(GEO)                                              CEL1  11
      ARI=1.                                                     CEL1  12
    4 AARI=ARI                                                   CEL1  13
      TEST=AARI*1.E-4                                            CEL1  14
      ARI=GEO+ARI                                                CEL1  15
C     TEST OF ACCURACY                                           CEL1  16
      IF(AARI-GEO-TEST)6,6,5                                     CEL1  17
    5 GEO=SQRT(AARI*GEO)                                         CEL1  18
      ARI=0.5*ARI                                                CEL1  19
      GO TO 4                                                    CEL1  20
    6 RES=3.1415927  /ARI                                        CEL1  21
      RETURN                                                     CEL1  22
      END                                                        CEL1  23
```

CEL2

This subroutine computes the generalized complete elliptic integral of the second kind. This is defined as

$$\text{cel } 2\ (k;\ A, B) = \int_0^{\pi/2} \frac{A + (B-A)\sin^2 t}{\sqrt{1 - k^2 \sin^2 t}}\ dt.$$

Equivalent is the definition:

$$\text{cel } 2\ (k;\ A, B) = \int_0^{\infty} \frac{A + Bx^2}{(1+x^2)\sqrt{(1+x^2)(1+k_c^2 x^2)}}\ dx,$$

where k_c is the complementary modulus:

$$k_c^2 + k^2 = 1,\ 0 < k_c^2 \leq 1$$

The subroutine CELI2 calculates cel 2 (k; A, B) for given modulus k, and constants A, B.

The calculation of RES = cel 2 (k, A, B) is based on the process of the Arithmetic-Geometric Mean.

Starting with the pair of numbers:

$$a_0 = 1,\ g_0 = k_c$$

the sequences of numbers (a_n), (g_n) are generated using for definition:

$$a_n = (a_{n-1} + g_{n-1}),\ g_n = 2\sqrt{a_{n-1}\, g_{n-1}}$$

This iteration process is stopped at the N^{th} step, when $a_N = g_N$.

Further needed are the sequences

(A_i), (B_i) defined by means of:

$$A_0 = A,\ B_0 = B$$

$$A_n = B_{n-1}/a_{n-1} + A_{n-1}$$

$$B_n = 2\,(B_{n-1} + g_{n-1} \cdot A_{n-1})$$

If D is the number of decimal digits in the mantissa of floating-point numbers, the iteration process is stopped as soon as $(a_{N-1} - g_{N-1})$ is less than $a_{N-1} \cdot 10^{-D/2}$.

Since (a_n), (g_n) converge quadratically to the same limit (Arithmetico-Geometrical mean) this implies that $(a_N - g_N)$ is less than $a_N \cdot 10^{-D}$.

The value of cel 2 (k; A, B) $= \frac{\pi}{4} \cdot \frac{A_{N+1}}{a_N}$

Subroutine CEL2

Purpose:

Computes the generalized complete elliptic integral of second kind.

Usage:

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

Description of parameters:

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

Remarks:

For AK = +1, -1, the result value is set 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 complementary modulus.
B(K) obtained with A = 1, B = 0.
D(K) obtained with A = 0, B = 1
where K, E, B, D define special cases of the generalized complete elliptic integral of second kind in the usual notation, and the argument K of these functions means the modulus.

Subroutines and function subprograms required:

None.

Method:

Definition:
RES= integral((A+B*T*T)/(SQRT((1+T*T)*(1+(CK*T)**2))*(1+T*T)) summed over T from 0 to infinity).
Evaluation:
Landen's transformation is used for calculation.
Reference:
R. Bulirsch, 'Numerical Calculation of Elliptic Integrals and Elliptic Functions', Handbook Series Special Functions, Numerische Mathematik Vol. 7, 1965, pp. 78-90.

```
      SUBROUTINE CEL2(RES,AK,A,B,IER)                                CEL2   1
      IER=0                                                          CEL2   2
C     TEST MODULUS                                                   CEL2   3
      GEO=1.-AK*AK                                                   CEL2   4
      IF(GEO)1,2,6                                                   CEL2   5
    1 IER=1                                                          CEL2   6
      RETURN                                                         CEL2   7
C     SET RESULT VALUE = OVERFLOW                                    CEL2   8
    2 IF(B)3,5,4                                                     CEL2   9
    3 RES=-1.E38                                                     CEL2  10
      RETURN                                                         CEL2  11
    4 RES=1.E38                                                      CEL2  12
      RETURN                                                         CEL2  13
    5 RES=A                                                          CEL2  14
      RETURN                                                         CEL2  15
C     COMPUTE INTEGRAL                                               CEL2  16
    6 GEO=SQRT(GEO)                                                  CEL2  17
      ARI=1.                                                         CEL2  18
      AA=A                                                           CEL2  19
      AN=A+B                                                         CEL2  20
      W=B                                                            CEL2  21
    7 W=W+AA*GEO                                                     CEL2  22
      W=W+W                                                          CEL2  23
      AA=AN                                                          CEL2  24
      AARI=ARI                                                       CEL2  25
      ARI=GEO+ARI                                                    CEL2  26
      AN=W/ARI+AN                                                    CEL2  27
C     TEST OF ACCURACY                                               CEL2  28
      IF(AARI-GEO-1.E-4*AARI)9,9,8                                   CEL2  29
    8 GEO=SQRT(GEO*AARI)                                             CEL2  30
      GEO=GEO+GEO                                                    CEL2  31
      GO TO 7                                                        CEL2  32
    9 RES=.7853982 *AN/ARI                                           CEL2  33
      RETURN                                                         CEL2  34
      END                                                            CEL2  35
```

EXPI

This subroutine computes the exponential integral in the range from -4 to infinity.

For positive x, the exponential integral is defined as:

$$E_1(x) = \int_x^\infty \frac{e^{-t}}{t}\,dt, \quad x > 0$$

This function, $E_1(x)$, may be analytically continued throughout the complex plane, and defines a multivalued complex function. However, for any given real argument, this extended multivalued function has a unique real part. The subroutine EXPI computes this unique real number for $x \geq -4$, $x \neq 0$.

For negative x, the real part of the extended exponential integral function is equal to $-E_i(-x)$,

where

$$E_i(y) = -\int_{-y}^\infty \frac{e^{-t}}{t}\,dt, \quad y > 0$$

($\int$ denotes Cauchy principal value.)

For x = 0, a singularity of the function, the program returns 1.0×10^{38}.

No action is taken in case of an argument less than -4.

Polynomial approximations which are close to Chebyshev approximations over their respective ranges are used for calculation.

1. Approximation in the range $x \geq 4$.

A polynomial approximation is obtained by means of truncation of the Expansion of $E_1(x)$ in terms of shifted Chebyshev Polynomials T_n*

$$E_1(x) = \frac{e^{-x}}{x} \sum_{n=0}^{\infty} A_n\, T_n^*\left(\frac{4}{x}\right), \quad \text{for } 4 \leq x < \infty$$

The coefficients A_n are given in the article by Luke/Wimp. *

Using only nine terms of the above infinite series results in a truncation error $\epsilon(x)$ with:

$$\left|\epsilon(x)\right| < \frac{e^{-x}}{x} \sum_{v=9}^{\infty} \left| A_v \right| < \frac{e^{-x}}{x} \cdot 0.82 \cdot 10^{-8}$$

Transformation of the shifted Chebyshev polynominals to ordinary polynomials finally leads to the approximation:

$$EXPI(x) = e^{-x} \left(\frac{4}{x}\right) \sum_{v=0}^{\infty} a_v \left(\frac{4}{x}\right)^v \quad \text{for } x \geq 4$$

The coefficients of this approximation given to eight signification digits are:

$a_0 = 0.24999\ 999$

$a_1 = -0.06249\ 8588$

$a_2 = 0.03120\ 8561$

$a_3 = -0.02295\ 1979$

$a_4 = 0.02041\ 2099$

$a_5 = -0.01755\ 5779$

$a_6 = 0.01172\ 3273$

$a_7 = -0.00493\ 62007$

$a_8 = 0.00094\ 42761\ 4$

2. Approximation in the range $|x| \leq 4$.

A polynomial approximation is obtained by means of telescoping of the Taylor series of the function:

$$\int_0^x \frac{(e^{-t}-1)}{t}\,dt = -\ln x - C - E_1(x),$$

where C = 0.57721 56649 is Euler's constant.

This results in the approximation:

$$EXPI(x) = -\ln|x| + \sum_{v=0}^{14} b_v x^v$$

---

*Luke/Wimp, "Jacobi Polynomial expansion of a generalized hypergeometric function over a semi-infinite ray", Math. Comp., Vol. 17, 1963, Iss. 84, p. 400.

with a truncation error E absolutely less than 3×10^{-8}.

The coefficients of this approximation given to eight significant digits are:

$b_0 = -0.57721\ 566$

$b_1 = 1.00000\ 00$

$b_2 = -0.25000\ 000$

$b_3 = 0.05555\ 5520$

$b_4 = -0.01041\ 6662$

$b_5 = 0.00166\ 66906$

$b_6 = -0.00023\ 14839\ 2$

$b_7 = 0.00002\ 83375\ 90$

$b_8 = -0.00000\ 30996\ 040$

$b_9 = 0.00000\ 03072\ 6221$

$b_{10} = -0.00000\ 00276\ 35830$

$b_{11} = 0.00000\ 00021\ 91569\ 9$

$b_{12} = -0.00000\ 00001\ 68265\ 92$

$b_{13} = 0.00000\ 00000\ 15798\ 675$

$b_{14} = -0.00000\ 00000\ 01031\ 7602$

Subroutine EXPI

Purpose:

Computes the exponential integral in the range -4 to infinity.

Usage:

CALL EXPI(RES, X, IER)

Description of parameters:

RES - Result value.
X - Argument of exponential integral.
IER - Resultant error code where:
IER=0 No error.
IER=1 X less than -4.

Remarks:

For X = 0 the result value is set to 1.E38.
For X less than -4 calculation is bypassed.
The argument remains unchanged.

Subroutines and function subprograms required:

None.

Method:

Definition:
RES= integral(EXP(-T)/T, summed over T from X to infinity).
Evaluation:
Two different polynomial approximations are used for X greater than 4 and for ABS(X) equal or less than 4.
Reference:
Luke and Wimp, 'Jacobi Polynomial Expansions of a Generalized Hypergeometric Function over a Semi-Infinite Range', Mathematical Tables and Other Aids to Computation, Vol. 17, 1963, Issue 84, pp. 395-404.

```
      SUBROUTINE EXPI(RES,X,IER)                                        EXPI   1
C        TEST OF RANGE                                                  EXPI   2
      IER=0                                                             EXPI   3
      IF(X-4.) 10,10,20                                                 EXPI   4
   10 IF(X+4.) 55,30,30                                                 EXPI   5
C        ARGUMENT IS GREATER THAN 4                                     EXPI   6
   20 ARG=4./X                                                          EXPI   7
      RES=EXP(-X)*(((((((((.0009442761&*ARG-.00493620071*ARG+.011723273) EXPI   8
     1 *ARG-.0175557791*ARG+.020412099)*ARG-.0229519791*ARG+.031208561) EXPI   9
     2 *ARG-.0624985881*ARG+.24999991)*ARG                              EXPI  10
      RETURN                                                            EXPI  11
C        ARGUMENT IS ABSOLUTELY LESS OR EQUAL 4                         EXPI  12
   30 IF(X) 40,50,40                                                    EXPI  13
   40RES=-ALOG(ABS(X))-(((((((((((((.10317602E-11*X-.15798675E-10)*X+   EXPI  14
     1.16826592E-9)*X-.21915699E-8)*X+.27635830E-7)*X-.30726221E-6)*X+  EXPI  15
     2.30996040E-5)*X-.28337590E-4)*X+.23148392E-3)*X-.0016666906)*X+   EXPI  16
     3.0104166621*X-.055555520)*X+.251*X-1.01*X-.57721566               EXPI  17
      RETURN                                                            EXPI  18
   50 RES=1.E38                                                         EXPI  19
      RETURN                                                            EXPI  20
C        ARGUMENT IS LESS THAN -4.                                      EXPI  21
   55 IER=1                                                             EXPI  22
      RETURN                                                            EXPI  23
      END                                                               EXPI  24
```

SICI

This subroutine computes the sine and cosine integrals. These integrals are defined as:

$$Si(x) = \int_{\infty}^{x} \frac{\sin(t)}{t}\, dt,\ x \geq 0$$

and

$$Ci(x) = \int_{\infty}^{x} \frac{\cos(t)}{t}\, dt,\ x > 0$$

The subroutine SICI calculates both Si(x) and Ci(x) for a given argument x. Two different approximations are used for the ranges $|x| \leq 4$ and $4 < |x| < \infty$. Negative values of the argument x are handled by means of the following symmetries:

$$Si(-x) = -\pi - Si(x)$$

Real part of

$$Ci(-x) = Ci(x),\ x > 0 \text{ (see discussion of EXPI).}$$

For x = 0, a singularity of Ci(x), the routine returns -1.0×10^{38}.

Polynomial approximations that are close to Chebyshev approximations over their respective ranges are used for calculation.

1. Approximation in the range $|x| > 4$.

The sine and cosine integrals are closely related to the confluent hypergeometric function:

$$Y(x) = -ix\, \Psi(1, 1; -ix).$$

We have:

$$Si(x) + i\, Ci(x) = \frac{\pi}{2} + ie^{ix}\, \Psi(1, 1; -ix).$$

Setting:

$$ix\, \Psi(1, 1; ix) = \sum_{n=0}^{\infty} (A_n + i B_n)\, T_n^* \left(\frac{4}{x}\right)$$

we get the expansions:

$$Si(x) = \sum_{n=0}^{\infty} \left(\frac{A_n \cdot \cos x}{x} + \frac{B_n \cdot \sin x}{x} \right) T_n^* \left(\frac{4}{x}\right)$$

$$Ci(x) = \sum_{n=0}^{\infty} \left(\frac{B_n \cdot \cos x}{x} - \frac{A_n \cdot \sin x}{x} \right) T_n \left(\frac{4}{x}\right)$$

in terms of shifted Chebyshev polynomials T_n^*.

The coefficients A_n and B_n are given in the article by Luke/Wimp.*

Using only ten terms of the above infinite series results in a truncation error E(x) with:

$$\left|E(x)\right| < \frac{1}{x} \cdot 2.3 \cdot 10^{-8}$$

Transformation of the shifted Chebyshev polynomials to ordinary polynomials finally leads to the approximations:

$$Si(x) = -\left(\frac{4}{x}\right) \cdot (\cos x \cdot V(x) + \sin x \cdot U(x))$$

$$Ci(x) = \left(\frac{4}{x}\right) \cdot (\sin x \cdot V(x) - \cos x \cdot U(x)),$$

where

$$V(x) = \sum_{n=0}^{10} a_n \cdot \left(\frac{4}{x}\right)^n$$

$$U(x) = \sum_{n=0}^{9} b_n \cdot \left(\frac{4}{x}\right)^n$$

The coefficients of these expansions given to eight significant digits are:

a_0 = 0.25000 000

b_0 = 0.00000 00002 58398 86

a_1 = -0.00000 06646 4406

b_1 = 0.06250 0111

a_2 = -0.03122 4178

b_2 = -0.00001 13495 79

*Luke/Wimp, "Jacobi Polynomial expansion of a generalized hypergeometric function over a semi-infinite ray", Math. Comp. Vol. 17, 1963, Iss. 84, p. 402.

$a_3 = -0.00037\ 64000\ 3$

$b_3 = -0.02314\ 6168$

$a_4 = 0.02601\ 2930$

$b_4 = -0.00333\ 25186$

$a_5 = -0.00794\ 55563$

$b_5 = 0.04987\ 7159$

$a_6 = -0.04400\ 4155$

$b_6 = -0.07261\ 6418$

$a_7 = 0.07902\ 0335$

$b_7 = 0.05515\ 0700$

$a_8 = -0.06537\ 2834$

$b_8 = -0.02279\ 1426$

$a_9 = 0.02819\ 1786$

$b_9 = 0.00404\ 80690$

$a_{10} = -0.00510\ 86993$

2. Approximation in the range $|x| \leq 4$.

A polynomial approximation for Si (x) is obtained by means of telescoping of the Taylor series:

$$Si(x) = -\frac{\pi}{2} + \int_0^x \frac{\sin t}{t}\, dt$$

$$= -\frac{\pi}{2} + x \cdot \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n+1) \cdot (2n+1)!}$$

This results in the approximation:

$$Si(x) = -\frac{\pi}{2} + x \cdot \sum_{n=0}^{6} a_v (x^2)^v ,$$

with a truncation error E absolutely less than $|X| \cdot 1.4 \cdot 10^{-9}$.

Similarly an approximation for Ci (x) is obtained by means of telescoping of the Taylor series:

$$Ci(x) - C - \ln(x) = \sum_{n=1}^{\infty} \frac{(-1)^n x^{2n}}{2N \cdot (2n)!}$$

This results in the approximation:

$$Ci(x) = C + \ln|x| - x^2 \cdot \sum_{n=0}^{5} b_n (x^2)^n ,$$

with a truncation error E absolutely less than $x^2 \cdot 5.6 \cdot 10^{-9}$.

The coefficients of these approximations given to eight significant decimal digits are:

$C = 0.57721\ 566$

$a_0 = 1.00000\ 00$

$b_0 = 0.24999\ 999$

$a_1 = -0.05555\ 5547$

$b_1 = -0.01041\ 6642$

$a_2 = 0.00166\ 66582$

$b_2 = 0.00023\ 14630\ 3$

$a_3 = -0.00002\ 83414\ 60$

$b_3 = -0.00000\ 30952\ 207$

$a_4 = 0.00000\ 03056\ 1233$

$b_4 = 0.00000\ 00269\ 45842$

$a_5 = -0.00000\ 00022\ 23263\ 3$

$b_5 = -0.00000\ 00001\ 38698\ 51$

$a_6 = -0.00000\ 00000\ 09794\ 2154$

Subroutine SICI

Purpose:
Computes the sine and cosine integral.

Usage:
CALL SICI(SI, CI, X)

Description of parameters:
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.

Subroutines and function subprograms required:

None.

Method:

Definition:

SI(X)= integral (SIN(T)/T, summed over T from infinity to X).

CI(X)= integral (COS(T)/T, summed over T from infinity to X).

Evaluation:

Reduction of range using symmetry.

Different approximations are used for ABS(X) greater than 4 and for ABS(X) less than 4.

Reference:

Luke and Wimp, 'Polynomial Approximations to Integral Transforms', Mathematical Tables and Other Aids to Computation, Vol. 15, 1961, Issue 74, pp. 174-178.

```
      SUBROUTINE SICI(SI,CI,X)                                          SICI   1
C        TEST ARGUMENT RANGE                                            SICI   2
      Z=ABS(X)                                                          SICI   3
      IF(Z-4.) 10,10,50                                                 SICI   4
C        Z IS NOT GREATER THAN 4                                        SICI   5
   10 Y=Z*Z                                                             SICI   6
      SI=-1.57079632*Y*(((((.97942154E-11*Y-.22232633E-8)*Y+.30561733E-5SICI   7
     1)*Y-.28341460E-4)*Y+.16666582E-2)*Y-.55555547E-1)*Y+1.           SICI   8
C        TEST FOR LOGARITHMIC SINGULARITY                               SICI   9
      IF(Z) 30,20,30                                                    SICI  10
   20 CI=-1.E38                                                         SICI  11
      RETURN                                                            SICI  12
   30 CI=0.57721566+ALOG(Z)-Y*((((((-.13869851E-9*Y+.26945842E-7)*Y-    SICI  13
     1.30952207E-5)*Y+.23146303E-3)*Y-.10416642E-1)*Y+.24999999)        SICI  14
   40 RETURN                                                            SICI  15
C        Z IS GREATER THAN 4.                                           SICI  16
   50 SI=SIN(Z)                                                         SICI  17
      Y=COS(Z)                                                          SICI  18
      Z=4./Z                                                            SICI  19
      QU=((((((((.40480590E-2*Z-.02279426)*Z+.05515070)*Z-.07261641)*Z+SICI  20
     1+.04987715)*Z-.33325186E-2)*Z-.07314616)*Z-.11349579E-4)*Z       SICI  21
     2+.06259011)*Z+.25839886E-9                                        SICI  22
      QV=(((((((((-.00510869)*Z+.02919178)*Z-.06537283)*Z+.07902033)*SICI  23
     1Z-.04400455)*Z-.00794555)*Z+.02601293)*Z-.37640003E-3)*Z         SICI  24
     2-.03122478)*Z-.56464406E-6)*Z+.25000000                           SICI  25
      CI=Z*(SI*V-Y*U)                                                   SICI  26
      SI=-Z*(SI*U+Y*V)                                                  SICI  27
C        TEST FOR NEGATIVE ARGUMENT                                     SICI  28
      IF(X) 60,40,40                                                    SICI  29
C        X IS LESS THAN -4.                                             SICI  30
   60 SI=-3.1415927-SI                                                  SICI  31
      RETURN                                                            SICI  32
      END                                                               SICI  33
```

CS

This subroutine computes the Fresnel integrals for a given value of the argument x. The Fresnel integrals are defined as:

$$C(x) = \frac{1}{\sqrt{2\pi}} \int_0^x \frac{\cos(t)}{\sqrt{t}}\, dt$$

and

$$S(x) = \frac{1}{\sqrt{2\pi}} \int_0^x \frac{\sin(t)}{\sqrt{t}}\, dt.$$

The subroutine CS calculates both C(x) and S(x) for a given argument x.

In case of a negative argument x the absolute value of x is taken as argument for C and for S.

Polynomial approximations that are close to Chebyshev approximations over their respective ranges are used for calculation.

1. Approximation in the range $|x| > 4$.

The Fresnel integrals C(x) and S(x) are closely related to the confluent hypergeometric function:

$$Y(x) = \sqrt{xi}\ \psi\left(\frac{1}{2}, \frac{1}{2}; xi\right) = xi\ \psi\left(1, \frac{3}{2}; xi\right).$$

We have:

$$C(x) = \frac{1}{2} + \frac{1}{\sqrt{8\pi}} \sqrt{\frac{4}{x}} \left(\sin(x)\, \mathrm{Re}(Y) - \cos(x)\, \mathrm{Im}(Y)\right)$$

$$S(x) = \frac{1}{2} - \frac{1}{\sqrt{8\pi}} \sqrt{\frac{4}{x}} \left(\cos(x)\, \mathrm{Re}(Y) + \sin(x)\, \mathrm{Im}(Y)\right)$$

The expansions of real part Re (Y) and complex part Im (Y) in terms of shifted Chebyshev polynomials T_n^* over the range $4 \le x < \infty$ are easily obtained using the method of computation described by Luke/Wimp.*

By means of truncation of the infinite series:

$$\mathrm{Re}\ (Y\ (x)) = \sum_{v=0}^{\infty} A_v T_v^* \left(\frac{4}{x}\right)$$

$$\mathrm{Im}\ (Y\ (x)) = \sum_{v=0}^{\infty} B_v T_v^* \left(\frac{4}{x}\right)$$

*Luke/Wimp, "Jacobi Polynomial expansion of a generalized hypergeometric function over a semi-infinite ray", Math. Comp., Vol. 17, 1963, Iss. 84, pp. 395-404.

beyond the eighth and ninth term respectively we get approximations with errors $E_c(x)$ and $E_s(x)$ where both errors are absolutely less than:

$$\epsilon = \sqrt{\frac{4}{x}} \cdot 1.3 \cdot 10^{-8}$$

Transformation of the shifted Chebyshev polynomials to ordinary polynomials finally leads to the approximations:

$$C(x) = \frac{1}{2} + \sqrt{\frac{4}{x}} \left(\sin(x) \cdot P(x) + \cos(x) \cdot Q(x)\right)$$

$$S(x) = \frac{1}{2} + \sqrt{\frac{4}{x}} \left(-\cos(x) \cdot P(x) + \sin(x) \cdot Q(x)\right)$$

where

$$P(x) = \sum_{0}^{7} a_v \left(\frac{4}{x}\right)^v$$

$$Q(x) = \sum_{0}^{8} b_v \left(\frac{4}{x}\right)^v .$$

The coefficients a_v and b_v are given to eight significant decimal digits:

$a_0 = 0.19947\ 115$

$b_0 = -0.00000\ 00044\ 44090\ 9$

$a_1 = -0.00000\ 12079\ 984$

$b_1 = -0.02493\ 3215$

$a_2 = -0.00931\ 49105$

$b_2 = -0.00001\ 60642\ 81$

$a_3 = -0.00040\ 27145\ 0$

$b_3 = 0.00597\ 21508$

$a_4 = 0.00742\ 82459$

$b_4 = -0.00030\ 95341\ 2$

$a_5 = -0.00727\ 16901$

$b_5 = -0.00679\ 28011$

$a_6 = 0.00340\ 14090$

$b_6 = 0.00797\ 09430$

$a_7 = -0.00066\ 33925\ 6$

$b_7 = -0.00416\ 92894$

$b_8 = 0.00087\ 68258$

2. Approximation in the range $0 \leqq x \leqq 4$.

Approximations for C(x) and S(x) in the range $0 \leqq x \leqq 4$ were obtained by means of telescoping of the respective Taylor series expansions:

$$C(x) = \sqrt{\frac{2}{\pi}} \cdot \sqrt{x} \cdot \sum_{v=0}^{\infty} \frac{(-1)^v x^{2v}}{(4v+1)\,(2v)!}$$

$$S(x) = \sqrt{\frac{2}{\pi}} \cdot \sqrt{x^3} \cdot \sum_{v=0}^{\infty} \frac{(-1)^v x^{2v}}{(4v+3)\,(2v+1)!}$$

This leads finally to the following approximations:

$$C(x) = \sqrt{x} \sum_{v=0}^{6} c_v \cdot (x^2)^v$$

$$S(x) = x\sqrt{x} \sum_{v=0}^{5} d_v (x^2)^v ,$$

with respective errors $E_c(x)$ and $E_s(x)$, where

$$\left|E_c(x)\right| < \sqrt{x} \cdot 2.6 \cdot 10^{-8}$$

$$\left|E_s(x)\right| < x\sqrt{x} \cdot 3.5 \cdot 10^{-8}$$

The coefficients c_v and d_v are given below to eight significant decimal digits:

$c_0 = 0.79788\ 455$

$d_0 = 0.26596\ 149$

$c_1 = -0.07978\ 8405$

$d_1 = -0.01899\ 7110$

$c_2 = 0.00369\ 38586$

$d_2 = 0.00060\ 43537\ 1$

$c_3 = -0.00008\ 52246\ 22$

$d_3 = -0.00001\ 05258\ 53$

$c_4 = 0.00000\ 11605\ 284$

$d_4 = 0.00000\ 01122\ 5331$

$c_5 = -0.00000\ 00101\ 40729$

$d_5 = -0.00000\ 00006\ 67774\ 47$

$c_6 = 0.00000\ 00000\ 50998\ 348$

```
      SUBROUTINE CSIC,S,X)                                              CS     1
      Z=ABS(X)                                                          CS     2
    2 IF(Z-4.) 3,3,4                                                    CS     3
C       X IS NOT GREATER THAN 4                                         CS     4
    3 C=SQRT(Z)                                                         CS     5
      S=Z*C                                                             CS     6
      Z=Z*Z                                                             CS     7
      C=C*(((((((.50998348E-10*Z-.10140729E-7)*Z+.11605284E-5)*Z        CS     8
     1 -.85224622E-4)*Z+.36938586E-2)*Z-.079788405)*Z+.797884551)       CS     9
      S=S*(((((-.66777447E-9*Z+.11275331E-6)*Z-.10525853E-4)*Z          CS    10
     1 +.60435371E-3)*Z-.18997110E-1)*Z+.26596149)                      CS    11
      RETURN                                                            CS    12
C       X IS GREATER THAN 4                                             CS    13
    4 D=COS(Z)                                                          CS    14
      S=SIN(Z)                                                          CS    15
      Z=4./Z                                                            CS    16
      A=(((((((.87682593E-3*Z-.41692894E-2)*Z+.79709430E-2)*Z-          CS    17
     1.67928011F-2)*Z-.30953412E-3)*Z+.59721508E-2)*Z-.16064281E-4)*Z-  CS    18
     2.0249332151*Z-.44440909E-8                                        CS    19
      B=(((((-.66339256E-3*Z+.34014090E-2)*Z-.72716901E-2)*Z+           CS    20
     1.74282459E-2)*Z-.40271450E-3)*Z-.93149105E-2)*Z-.12079984E-5)*Z+  CS    21
     2.1994711                                                          CS    22
      Z=SQRT(Z)                                                         CS    23
      C=.5+Z*(D*A+S*B)                                                  CS    24
      S=.5+Z*(S*A-D*B)                                                  CS    25
      RETURN                                                            CS    26
      END                                                               CS    27
```

Subroutine CS

Purpose:

Computes the Fresnel integrals.

Usage:

CALL CS (C, S, X)

Description of parameters:

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.

Subroutines and function subprograms required:

None.

Method:

Definition:

C(X)= integral (COS(T)/SQRT(2*PI*T) summed over T from 0 to X).

S(X)= integral (SIN(T)/SQRT(2*PI*T) summed over T from 0 to X).

Evaluation:

Using different approximations for X less than 4 and X greater than 4.

Reference:

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

SIMQ

Purpose:

Obtain solution of a set of simultaneous linear equations, AX=B.

Usage:

CALL SIMQ(A,B,N,KS)

Description of parameters:

- A - Matrix of coefficients stored columnwise. 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 matrix is singular, solution values are meaningless.

An alternative solution may be obtained by using matrix inversion (MINV) and matrix product (GMPRD).

Subroutines and function subprograms required:

None.

Method:

Method of solution is by elimination using 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), ..., 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.

```
      SUBROUTINE SIMQ(A,B,N,KS)                                         SIMQ   1
      DIMENSION A(1),B(1)                                               SIMQ   2
C     FORWARD SOLUTION                                                  SIMQ   3
      TOL=0.0                                                           SIMQ   4
      KS=0                                                              SIMQ   5
      JJ=-N                                                             SIMQ   6
      DO 65 J=1,N                                                       SIMQ   7
      JY=J+1                                                            SIMQ   8
      JJ=JJ+N+1                                                         SIMQ   9
      BIGA=0                                                            SIMQ  10
      IT=JJ-J                                                           SIMQ  11
      DO 30 I=J,N                                                       SIMQ  12
C     SEARCH FOR MAXIMUM COEFFICIENT IN COLUMN                          SIMQ  13
      IJ=IT+I                                                           SIMQ  14
      IF(ABS(BIGA)-ABS(A(IJ))) 20,30,30                                 SIMQ  15
   20 BIGA=A(IJ)                                                        SIMQ  16
      IMAX=I                                                            SIMQ  17
   30 CONTINUE                                                          SIMQ  18
C     TEST FOR PIVOT LESS THAN TOLERANCE (SINGULAR MATRIX)              SIMQ  19
      IF(ABS(BIGA)-TOL) 35,35,40                                        SIMQ  20
   35 KS=1                                                              SIMQ  21
      RETURN                                                            SIMQ  22
C     INTERCHANGE ROWS IF NECESSARY                                     SIMQ  23
   40 II=J+N*(J-2)                                                      SIMQ  24
      IT=IMAX-J                                                         SIMQ  25
      DO 50 K=J,N                                                       SIMQ  26
      II=II+N                                                           SIMQ  27
      I2=II+IT                                                          SIMQ  28
      SAVE=A(II)                                                        SIMQ  29
      A(II)=A(I2)                                                       SIMQ  30
      A(I2)=SAVE                                                        SIMQ  31
C     DIVIDE EQUATION BY LEADING COEFFICIENT                            SIMQ  32
   50 A(II)=A(II)/BIGA                                                  SIMQ  33
      SAVE=B(IMAX)                                                      SIMQ  34
      B(IMAX)=B(J)                                                      SIMQ  35
      B(J)=SAVE/BIGA                                                    SIMQ  36
C     ELIMINATE NEXT VARIABLE                                           SIMQ  37
      IF(J-N) 55,70,55                                                  SIMQ  38
   55 IQS=N*(J-1)                                                       SIMQ  39
      DO 65 IX=JY,N                                                     SIMQ  40
      IXJ=IQS+IX                                                        SIMQ  41
      IT=J-IX                                                           SIMQ  42
      DO 60 JX=JY,N                                                     SIMQ  43
      IXJX=N*(JX-1)+IX                                                  SIMQ  44
      JJX=IXJX+IT                                                       SIMQ  45
   60 A(IXJX)=A(IXJX)-(A(IXJ)*A(JJX))                                   SIMQ  46
   65 B(IX)=B(IX)-(B(J)*A(IXJ))                                         SIMQ  47
C     BACK SOLUTION                                                     SIMQ  48
   70 NY=N-1                                                            SIMQ  49
      IT=N*N                                                            SIMQ  50
      DO 80 J=1,NY                                                      SIMQ  51
      IA=IT-J                                                           SIMQ  52
      IB=N-1                                                            SIMQ  53
      IC=1                                                              SIMQ  54
      DO 80 K=1,J                                                       SIMQ  55
      B(IB)=B(IB)-A(IA)*B(IC)                                           SIMQ  56
      IA=IA-N                                                           SIMQ  57
   80 IC=IC-1                                                           SIMQ  58
      RETURN                                                            SIMQ  59
      END                                                               SIMQ  60
```

RTWI

This subroutine refines the initial guess x_0 of a root of the general nonlinear equation $x = f(x)$. Wegstein's iteration scheme is used in order to get accelerated convergence in case of a function f (x), which has at least continuous first derivative in the range in which iteration moves.

Following Figure 8, set $x_1 = y_0 = f(x_0)$ and $y_1 = f(x_1)$.

Refinement of x_1 is done by determination of the intersection of the linear function $y = x$ and the secant through the points (x_0, y_0) and (x_1, y_1), thus getting:

$$x_2 = x_1 + \frac{x_1 - x_0}{\frac{x_0 - y_0}{x_1 - y_1} - 1}$$

and $y_2 = f(x_2)$

The next step is done by starting at (x_2, y_2) and setting:

$$x_3 = x_2 + \frac{x_2 - x_1}{\frac{x_1 - y_1}{x_2 - y_2} - 1}$$

$$y_3 = f(x_3)$$

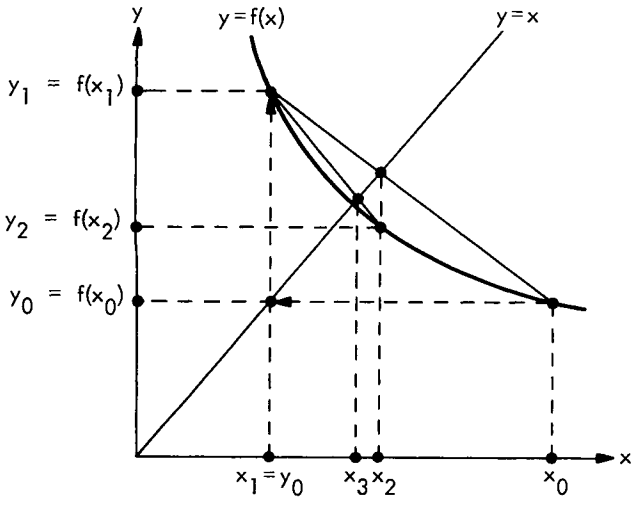


Figure 8. Wegstein's iterative method

It can be seen that this determines the intersection between $y = x$ and the secant through the points (x_1, y_1) and (x_2, y_2). Therefore Wegstein's iteration scheme is often called the secant modification of the normal iteration scheme $x_{i+1} = f(x_i)$.

Repeating these steps, the result is the iteration scheme:

$$\left.\begin{aligned} x_{i+1} &= x_i + \frac{x_i - x_{i-1}}{\frac{x_{i-1} - y_{i-1}}{x_i - y_i} - 1} \\ y_{i+1} &= f(x_{i+1}) \end{aligned}\right\} (i = 1, 2, \ldots) \quad (1)$$

Each step requires one evaluation of f(x).

This iterative procedure is terminated if the following two conditions are satisfied:

$$\delta_1 \leqq \text{and} \quad \delta_2 \leqq 10 \cdot \varepsilon$$

with

$$\delta_1 = \begin{cases} \left|\frac{x_{i+1} - x_i}{x_{i+1}}\right| & \text{if} \quad |x_{i+1}| > 1 \\ |x_{i+1} - x_i| & \text{if} \quad |x_{i+1}| \leqq 1, \end{cases} \quad (2)$$

$$\delta_2 = \begin{cases} \left|\frac{x_{i+1} - y_{i+1}}{x_{i+1}}\right| & \text{if} \quad |x_{i+1}| > 1 \\ |x_{i+1} - y_{i+1}| & \text{if} \quad |x_{i+1}| \leqq 1 \end{cases}$$

and tolerance ε given by input.

The procedure described above may not converge within a specified number of iteration steps. Reasons for this behavior, which is indicated by an error message may be:

1. Too few iteration steps are specified.
2. The initial guess x_0 is too far away from any root.
3. The tolerance ε is too small with respect to roundoff errors.
4. The root to be determined is of multiplicity greater than one.

Furthermore, the procedure fails if at any iteration step the denominator of equation (1) becomes zero. This is also indicated by an error message. This failure may have two reasons:

1. The secant has the slope 1, either exactly or due to roundoff errors. In both cases it is probable that there is at least one point ξ in the range in which iteration moves with $f'(\xi) = 1$.
2. $x_i = x_{i-1}$ and $x_i \neq y_i = f(x_i)$. This case is possible due to roundoff errors or to a very steep slope of the secant.

Subroutine RTWI

Purpose:

To solve general nonlinear equations of the form X=FCT(X) by means of Wegstein's iteration method.

Usage:

CALL RTWI (X, VAL, FCT, XST, EPS, IEND, IER)
Parameter FCT requires an EXTERNAL statement.

Description of parameters:

- X - Resultant root of equation X=FCT(X).
- VAL - Resultant value of X-FCT(X) at root X.
- FCT - Name of the external function subprogram used.
- XST - Input value which specifies the initial guess of the root X.
- EPS - Input value which specifies the upper bound of the error of result X.
- IEND - Maximum number of iteration steps specified.
- IER - Resultant error parameter coded as follows:
  - IER=0 - no error
  - IER=1 - no convergence after IEND iteration steps
  - IER=2 - at some iteration step the denominator of iteration formula was equal to zero

Remarks:

The procedure is bypassed and gives the error message IER=2 if at any iteration steps the denominator of the iteration formula is equal to zero. That means that there is at least one point in the range in which iteration moves with the derivative of FCT(X) equal to 1.

Subroutines and function subprograms required:

The external function subprogram FCT(X) must be furnished by the user.

Method:

Solution of equation X=FCT(X) is done by means of Wegstein's iteration method, which starts at the initial guess XST of a root X. One iteration step requires one evaluation of FCT(X). For test on satisfactory accuracy see formula (2) of mathematical description.

For reference, see:

1. G. N. Lance, Numerical Methods for High Speed Computers, Iliffe, London, 1960, pp. 134-138.
2. J. Wegstein, "Algorithm 2," CACM, Vol. 3, Iss. 2 (1960), pp. 74.
3. H.C. Thacher, "Algorithm 15," CACM, Vol. 3, Iss. 8 (1960), pp. 475.
4. J.G. Herriot, "Algorithm 26," CACM, Vol. 3, Iss. 11 (1960), pp. 603.


```
      SUBROUTINE RTWI(X,VAL,FCT,XST,EPS,IEND,IER)                       RTWI   1
C     PREPARE ITERATION                                                 RTWI   2
      IER=0                                                             RTWI   3
      TOL=XST                                                           RTWI   4
      X=FCT(TOL)                                                        RTWI   5
      A=X-XST                                                           RTWI   6
      B=-A                                                              RTWI   7
      TOL=X                                                             RTWI   8
      VAL=X-FCT(TOL)                                                    RTWI   9
C     START ITERATION LOOP                                              RTWI  10
      DO 6 I=1,IEND                                                     RTWI  11
      IF(VAL)1,7,1                                                      RTWI  12
C     EQUATION IS NOT SATISFIED BY X                                    RTWI  13
    1 B=B/VAL-1.                                                        RTWI  14
      IF(B)2,8,2                                                        RTWI  15
C     ITERATION IS POSSIBLE                                             RTWI  16
    2 A=A/B                                                             RTWI  17
      X=X+A                                                             RTWI  18
      B=VAL                                                             RTWI  19
      TOL=X                                                             RTWI  20
      VAL=X-FCT(TOL)                                                    RTWI  21
C     TEST ON SATISFACTORY ACCURACY                                     RTWI  22
      TOL=EPS                                                           RTWI  23
      D=ABS(X)                                                          RTWI  24
      IF(D-1.)6,4,3                                                     RTWI  25
    3 TOL=TOL*D                                                         RTWI  26
    4 IF(ABS(A)-TOL)5,5,6                                               RTWI  27
    5 IF(ABS(VAL)-10.*TOL)7,7,6                                         RTWI  28
    6 CONTINUE                                                          RTWI  29
C     END OF ITERATION LOOP                                             RTWI  30
C     NO CONVERGENCE AFTER IEND ITERATION STEPS. ERROR RETURN.          RTWI  31
      IER=1                                                             RTWI  32
    7 RETURN                                                            RTWI  33
C     ERROR RETURN IN CASE OF ZERO DIVISOR                              RTWI  34
    8 IER=2                                                             RTWI  35
      RETURN                                                            RTWI  36
      END                                                               RTWI  37
```

RTMI

This subroutine determines a root of the general nonlinear equation $f(x) = 0$ in the range of x from x_{li} up to x_{ri} (x_{li}, x_{ri} given by input) by means of Mueller's iteration scheme of successive bisection and inverse parabolic interpolation. The procedure assumes $f(x_{li}) \cdot f(x_{ri}) \leqq 0$.

Starting with $x_l = x_{li}$ and $x_r = x_{ri}$ and following Fig. 9, one iteration step is described.

First, the middle of the interval $x_l \ldots x_r$ is computed:

$$x_m = \frac{1}{2}(x_l + x_r).$$

In case $f(x_m) \cdot f(x_r) < 0$, x_l and x_r are interchanged to ensure that $f(x_m) \cdot f(x_r) > 0$.

In case

$$2 f(x_m)\left[f(x_m)-f(x_l)\right] - f(x_r)\left[f(x_r) - f(x_l)\right] \geqq 0 \quad (1)$$

x_r is replaced by x_m and the bisection step is repeated. If, after a specified number of successive bisections, inequality (1) is still satisfied, the procedure is bypassed and an error message is given.

In Fig. 9, the second bisection step leads to a configuration which does not satisfy inequality (1). Thus by inverse parabolic interpolation:

$$\Delta x = f(x_l)\,\frac{x_m - x_l}{f(x_m)-f(x_l)} \left\{1 + f(x_m)\,\frac{f(x_r) - 2f(x_m)+f(x_l)}{\left[f(x_r)-f(x_m)\right]\left[f(x_r)-f(x_l)\right]}\right\} \quad (2)$$

and $x = x_l - \Delta x$

and x is sure to be situated between x_l and x_m

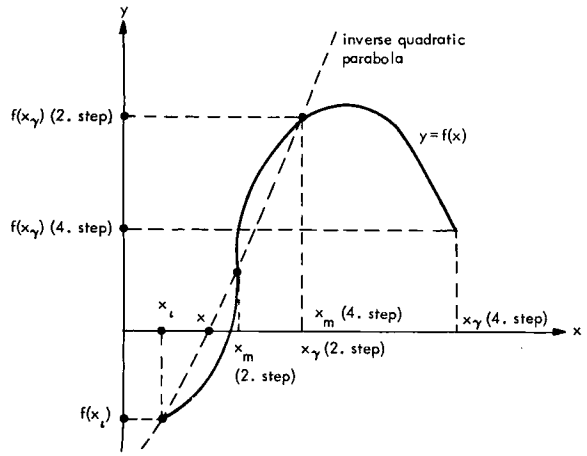


Figure 9. Mueller's iterative method

Now, for the next iteration step, x becomes x_l and x_m becomes x_r if $f(x) \cdot f(x_l) > 0$, or x becomes x_r if $f(x) \cdot f(x_l) \leq 0$.

Convergence is either quadratic or linear if the multiplicity of the root to be determined is equal to one or greater than one respectively, and if f(x) can be differentiated continuously at least twice in the range $x_{li} \cdots x_{ri}$. Each iteration step requires two evaluations of f(x).

This iterative procedure is terminated if either the two conditions (checked in bisection loop)

$$\left.\begin{aligned} &|x_r - x_l| \leq \varepsilon \cdot \max(1, |x_r|) \\ \text{and} \quad &|f(x_r) - f(x_l)| \leq 100 \cdot \varepsilon \end{aligned}\right\} \qquad (3)$$

or the two conditions (checked after inverse parabolic interpolation)

$$\left.\begin{aligned} &|\Delta x| \leq \varepsilon \cdot \max(1, |x|) \\ \text{and} \quad &|f(x)| \leq 100 \cdot \varepsilon \end{aligned}\right\} \qquad (4)$$

are satisfied, where tolerance ε is given by input.

The procedure described above may not converge within a specified number of iteration steps followed by the same number of successive bisections. Reasons for this behaviour, which is indicated by an error message, may be:

1. Too few iteration steps are specified.
2. The initial interval $x_{li} \cdots x_{ri}$ is too long.
3. The tolerance ε is too small with respect to roundoff errors.

Furthermore, the procedure is bypassed, also giving an error message, if the basic assumption $f(x_{li}) \cdot f(x_{ri}) \leq 0$ is not satisfied.

For reference see G. K. Kristiansen, "Zero of Arbitrary Function", BIT, vol. 3 (1963), pp. 205-206.

Subroutine RTMI

Purpose:

To solve general nonlinear equations of the form FCT(X)=0 by means of Mueller's iteration method.

Usage:

CALL RTMI(X, F, FCT, XLI, XRI, EPS, IEND, IER)
Parameter FCT requires an EXTERNAL statement.

Description of parameters:

- 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 which specifies the initial left bound of the root X.
- XRI - Input value which specifies the initial right bound of the root X.
- EPS - Input value which specifies the upper bound of the error of result X.
- IEND - Maximum number of iteration steps specified.
- IER - Resultant error parameter coded as follows:
  - IER=0 - no error
  - IER=1 - no convergence after IEND iteration steps followed by IEND successive steps of bisection
  - IER=2 - basic assumption FCT(XLI) *FCT(XRI) less than or equal to zero is not satisfied

Remarks:

The procedure assumes that function values at initial bounds XLI and XRI have not the same sign. If this basic assumption is not satisfied by input values XLI and XRI, the procedure is bypassed and gives the error message IER=2.

Subroutines and function subprograms required:

The external function subprogram FCT(X) must be furnished by the user.

Method:

Solution of equation FCT(X)=0 is done by means of Mueller's iteration method of successive bisections and inverse parabolic interpolation, which 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). For test on satisfactory accuracy see formulae (3, 4) of mathematical description.

```
      SUBROUTINE RTMI(X,F,FCT,XL1,XR1,EPS,IEND,IER)                     RTMI   1
C     PREPARE ITERATION                                                 RTMI   2
      IER=0                                                             RTMI   3
      XL=XL1                                                            RTMI   4
      XR=XR1                                                            RTMI   5
      X=XL                                                              RTMI   6
      TOL=X                                                             RTMI   7
      F=FCT(TOL)                                                        RTMI   8
      IF(F)1,16,1                                                       RTMI   9
    1 FL=F                                                              RTMI  10
      X=XR                                                              RTMI  11
      TOL=X                                                             RTMI  12
      F=FCT(TOL)                                                        RTMI  13
      IF(F)2,16,2                                                       RTMI  14
    2 FR=F                                                              RTMI  15
      IF(SIGN(1.,FL)+SIGN(1.,FR))25,3,25                                RTMI  16
C     BASIC ASSUMPTION FL*FR LESS THAN 0 IS SATISFIED.                  RTMI  17
C     GENERATE TOLERANCE FOR FUNCTION VALUES.                           RTMI  18
    3 I=0                                                               RTMI  19
      TOLF=100.*EPS                                                     RTMI  20
C     START ITERATION LOOP                                              RTMI  21
    4 I=I+1                                                             RTMI  22
C     START BISECTION LOOP                                              RTMI  23
      DO 13 K=1,IEND                                                    RTMI  24
      X=.5*(XL+XR)                                                      RTMI  25
      TOL=X                                                             RTMI  26
      F=FCT(TOL)                                                        RTMI  27
      IF(F)5,16,5                                                       RTMI  28
    5 IF(SIGN(1.,F)+SIGN(1.,FR))7,6,7                                   RTMI  29
C     INTERCHANGE XL AND XR IN ORDER TO GET THE SAME SIGN IN F AND FR   RTMI  30
    6 TOL=XL                                                            RTMI  31
      XL=XR                                                             RTMI  32
      XR=TOL                                                            RTMI  33
      TOL=FL                                                            RTMI  34
      FL=FR                                                             RTMI  35
      FR=TOL                                                            RTMI  36
    7 TOL=F-FL                                                          RTMI  37
      A=F*TOL                                                           RTMI  38
      A=A+A                                                             RTMI  39
      IF(A-FR*(FR-FL))8,9,9                                             RTMI  40
    8 IF(I-IEND)17,17,9                                                 RTMI  41
    9 XR=X                                                              RTMI  42
      FR=F                                                              RTMI  43
C     TEST ON SATISFACTORY ACCURACY IN BISECTION LOOP                   RTMI  44
      TOL=EPS                                                           RTMI  45
      A=ABS(XR)                                                         RTMI  46
      IF(A-1.)11,11,10                                                  RTMI  47
   10 TOL=TOL*A                                                         RTMI  48
   11 IF(ABS(XR-XL)-TOL)12,12,13                                        RTMI  49
   12 IF(ABS(FR-FL)-TOLF)14,14,13                                       RTMI  50
   13 CONTINUE                                                          RTMI  51
C     END OF BISECTION LOOP                                             RTMI  52
C     NO CONVERGENCE AFTER IEND ITERATION STEPS FOLLOWED BY IEND        RTMI  53
C     SUCCESSIVE STEPS OF BISECTION OR STEADILY INCREASING FUNCTION     RTMI  54
C     VALUES AT RIGHT BOUNDS. ERROR RETURN.                             RTMI  55
      IER=1                                                             RTMI  56
   14 IF(ABS(FR)-ABS(FL))16,16,15                                       RTMI  57
   15 X=XL                                                              RTMI  58
      F=FL                                                              RTMI  59
   16 RETURN                                                            RTMI  60
C     COMPUTATION OF ITERATED X-VALUE BY INVERSE PARABOLIC INTERPOLATIONRTMI  61
   17 A=FR-F                                                            RTMI  62
      DX=(X-XL)*FL*(1.+F*(A-TOL)/(A*(FR-FL)))/TOL                       RTMI  63
      XM=X                                                              RTMI  64
      FM=F                                                              RTMI  65
      X=XL-DX                                                           RTMI  66
      TOL=X                                                             RTMI  67
      F=FCT(TOL)                                                        RTMI  68
      IF(F)18,16,18                                                     RTMI  69
C     TEST ON SATISFACTORY ACCURACY IN ITERATION LOOP                   RTMI  70
   18 TOL=EPS                                                           RTMI  71
      A=ABS(X)                                                          RTMI  72
      IF(A-1.)20,20,19                                                  RTMI  73
   19 TOL=TOL*A                                                         RTMI  74
   20 IF(ABS(DX)-TOL)21,21,22                                           RTMI  75
   21 IF(ABS(F)-TOLF)16,16,22                                           RTMI  76
C     PREPARATION OF NEXT BISECTION LOOP                                RTMI  77
   22 IF(SIGN(1.,F)+SIGN(1.,FL))24,23,24                                RTMI  78
   23 XR=X                                                              RTMI  79
      FR=F                                                              RTMI  80
      GO TO 4                                                           RTMI  81
   24 XL=X                                                              RTMI  82
      FL=F                                                              RTMI  83
      XR=XM                                                             RTMI  84
      FR=FM                                                             RTMI  85
      GO TO 4                                                           RTMI  86
C     END OF ITERATION LOOP                                             RTMI  87
C     ERROR RETURN IN CASE OF WRONG INPUT DATA                          RTMI  88
   25 IER=2                                                             RTMI  89
      RETURN                                                            RTMI  90
      END                                                               RTMI  91
```

RTNI

This subroutine refines the initial guess x_0 of a root of the general nonlinear equation f(x) = 0. Newton's iteration scheme is used in the following form:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \qquad (i = 0, 1, 2, \ldots) \qquad (1)$$

Convergence is quadratic or linear if the multiplicity of the root to be determined is equal to one or greater than one respectively, and if f(x) can be differentiated continuously at least twice in the range in which iteration moves. Each iteration step requires one evaluation of f(x) and one evaluation of $f'(x)$.

This iterative procedure is terminated if the following two conditions are satisfied:

$$\delta \leqq \varepsilon \text{ and } |f(x_{i+1})| \leqq 100 \cdot \varepsilon$$

with

$$\delta = \left\{ \begin{array}{l} \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| \text{ in case of } |x_{i+1}| > 1 \\ |x_{i+1} - x_i| \text{ in case of } |x_{i+1}| \leqq 1 \end{array} \right\} \qquad (2)$$

and tolerance ε given by input.

The procedure described above may not converge within a specified number of iteration steps. Reasons for this behaviour, which is indicated by an error message, may be:

1. Too few iteration steps are specified.
2. The initial guess x_0 is too far away from any root.
3. The tolerance ε is too small with respect to roundoff errors.
4. The root to be determined is of multiplicity greater than one.

Furthermore, the procedure fails and is bypassed if at any iteration step the derivative $f(x_i)$ becomes zero. This is also indicated by an error message.

For reference see:

(1) F. B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, New York/Toronto/London, 1956, pp. 447 - 450.

(2) R. Zurmühl, Praktische Mathematik für Ingenieure und Physiker, Springer, Berlin/Göttingen/Heidelberg, 1963, pp. 12 - 17.

Subroutine RTNI

Purpose:

To solve general nonlinear equations of the form F(X)=0 by means of Newton's iteration method.

Usage:

CALL RTNI (X, F, DERF, FCT, XST, EPS, IEND, IER) Parameter FCT requires an EXTERNAL statement

Description of parameters:

- 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 the external subroutine used. It computes for given argument X the function value F and derivative DERF. Its parameter list must be X, F, DERF.
- XST - Input value which specifies the initial guess of the root X.
- EPS - Input value which specifies the upper bound of the error of result X.

IEND - Maximum number of iteration steps specified.

IER - Resultant error parameter coded as follows:

IER=0 - no error

IER=1 - no convergence after IEND iteration steps

IER=2 - at some iteration step derivative DERF was equal to zero

Remarks:

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 were started again with another initial guess XST.

Subroutines and function subprograms required:

The external subroutine FCT(X, F, DERF) must be furnished by the user.

Method:

Solution of the equation F(X)=0 is obtained by means of Newton's iteration method, which 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). For tests on satisfactory accuracy see formula (2) of the mathematical description.

```
      SUBROUTINE RTNI(X,F,DERF,FCT,XST,EPS,IEND,IER)                    RTNI   1
C     PREPARE ITERATION                                                 RTNI   2
      IER=0                                                             RTNI   3
      X=XST                                                             RTNI   4
      TOL=X                                                             RTNI   5
      CALL FCT(TOL,F,DERF)                                              RTNI   6
      TOLF=100.*EPS                                                     RTNI   7
C     START ITERATION LOOP                                              RTNI   8
      DO 6 I=1,IEND                                                     RTNI   9
      IF(F)1,7,1                                                        RTNI  10
C     EQUATION IS NOT SATISFIED BY X                                    RTNI  11
    1 IF(DERF)2,8,2                                                     RTNI  12
C     ITERATION IS POSSIBLE                                             RTNI  13
    2 DX=F/DERF                                                         RTNI  14
      X=X-DX                                                            RTNI  15
      TOL=X                                                             RTNI  16
      CALL FCT(TOL,F,DERF)                                              RTNI  17
C     TEST ON SATISFACTORY ACCURACY                                     RTNI  18
      TOL=EPS                                                           RTNI  19
      A=ABS(X)                                                          RTNI  20
      IF(A-1.)4,4,3                                                     RTNI  21
    3 TOL=TOL*A                                                         RTNI  22
    4 IF(ABS(DX)-TOL)5,5,6                                              RTNI  23
    5 IF(ABS(F)-TOLF)7,7,6                                              RTNI  24
    6 CONTINUE                                                          RTNI  25
C     END OF ITERATION LOOP                                             RTNI  26
C     NO CONVERGENCE AFTER IEND ITERATION STEPS. ERROR RETURN.          RTNI  27
      IER=1                                                             RTNI  28
    7 RETURN                                                            RTNI  29
C     ERROR RETURN IN CASE OF ZERO DIVISOR                              RTNI  30
    8 IER=2                                                             RTNI  31
      RETURN                                                            RTNI  32
      END                                                               RTNI  33
```

Mathematics - Roots of Polynomial

POLRT

This subroutine computes the real and complex roots of a real polynomial.

Given a polynomial

$$f(z) = \sum_{n=0}^{N} a_n z^n \tag{1}$$

let

$Z = X + iY$ be a starting value for a root of f(z).

Then:

$$Z^n = (X + iY)^n. \tag{2}$$

Define X_n as real terms of expanded equation (2).

Define Y_n as imaginary terms of expanded equation (2).

Then for:

$n = 0$

$X_o = 1.0$

$Y_o = 0.0$

$n > 0$

$$X_n = X \cdot X_{n-1} - Y \cdot Y_{n-1} \tag{3}$$

$$Y_n = X \cdot Y_{n-1} + Y \cdot X_{n-1} \tag{4}$$

Let U be the real terms of (1).

V be the imaginary terms of (1).

Then:

$$U = \sum_{n=0}^{N} a_n X_n \tag{5}$$

$$V = \sum_{n=0}^{N} a_n Y_n \tag{6}$$

or

$$U = a_o + \sum_{n=1}^{N} a_n X_n \tag{7}$$

$$V = \sum_{n=1}^{N} a_n Y_n \tag{8}$$

$$\frac{\partial U}{\partial X} = \sum_{n=1}^{N} n \cdot X_{n-1} \cdot a_n \tag{9}$$

$$\frac{\partial U}{\partial Y} = -\sum_{n=1}^{N} n\, Y_{n-1}\, a_n \tag{10}$$

Note that equations (3), (4), (7), (8), (9), and (10) can be performed iteratively for n = 1 to N by saving X_{n-1} and Y_{n-1}.

Using the Newton-Raphson method for computing ΔX, ΔY, we have:

$$\Delta X = \left(V\frac{\partial U}{\partial Y} - U\frac{\partial U}{\partial X}\right) \Bigg/ \left[\left(\frac{\partial U}{\partial X}\right)^2 + \left(\frac{\partial U}{\partial Y}\right)^2\right] \quad (11)$$

$$\Delta Y = -\left(U\frac{\partial U}{\partial Y} + V\frac{\partial U}{\partial X}\right) \Bigg/ \left[\left(\frac{\partial U}{\partial X}\right)^2 + \left(\frac{\partial U}{\partial Y}\right)^2\right] \quad (12)$$

after applying the Cauchy-Riemann equations.

Thus, for the next iteration:

$$X' = X + \Delta X$$

$$Y' = Y + \Delta Y$$

Subroutine POLRT

Purpose:

Computes the real and complex roots of a real polynomial.

Usage:

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

Description of parameters:

- 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 - Error code where:
  - IER=0 No error.
  - IER=1 M less than one.
  - IER=2 M greater than 36.
  - IER=3 Unable to determine root with 500 iterations on 5 starting values.
  - IER=4 High order coefficient is zero.

Remarks:

Limited to 36th order polynomial or less. Floating-point overflow may occur for high order polynomials but will not affect the accuracy of the results.

Subroutines and function subprograms required:

None.

Method:

Newton-Raphson iterative technique. The final iterations on each root are performed using the original polynomial rather than the reduced polynomial to avoid accumulated errors in the reduced polynomial.

```
      SUBROUTINE POLRT(XCOF,COF,M,ROOTR,ROOTI,IER)                      POLRT  1
      DIMENSION XCOF(1),COF(1),ROOTR(1),ROOTI(1)                        POLRT  2
      IFIT=0                                                            POLRT  3
      N=M                                                               POLRT  4
      IER=0                                                             POLRT  5
      IF(XCOF(N+1)) 10,25,10                                            POLRT  6
   10 IF(N) 15,15,32                                                    POLRT  7
C        SET ERROR CODE TO 1                                            POLRT  8
   15 IER=1                                                             POLRT  9
   20 RETURN                                                            POLRT 10
C        SET ERROR CODE TO 4                                            POLRT 11
   25 IER=4                                                             POLRT 12
      GO TO 20                                                          POLRT 13
C        SET ERROR CODE TO 2                                            POLRT 14
   30 IER=2                                                             POLRT 15
      GO TO 20                                                          POLRT 16
   32 IF(N-36) 35,35,30                                                 POLRT 17
   35 NX=N                                                              POLRT 18
      NXX=N+1                                                           POLRT 19
      N2=1                                                              POLRT 20
      KJ1 = N+1                                                         POLRT 21
      DO 40 L=1,KJ1                                                     POLRT 22
      MT=KJ1-L+1                                                        POLRT 23
   40 COF(MT)=XCOF(L)                                                   POLRT 24
C        SET INITIAL VALUES                                             POLRT 25
   45 XO=.00500101                                                      POLRT 26
      YO=0.01000101                                                     POLRT 27
C        ZERO INITIAL VALUE COUNTER                                     POLRT 28
      IN=0                                                              POLRT 29
   50 X=XO                                                              POLRT 30
C        INCREMENT INITIAL VALUES AND COUNTER                           POLRT 31
      XO=-10.0*YO                                                       POLRT 32
      YO=-10.0*X                                                        POLRT 33
C        SET X AND Y TO CURRENT VALUE                                   POLRT 34
      X=XO                                                              POLRT 35
      Y=YO                                                              POLRT 36
      IN=IN+1                                                           POLRT 37
      GO TO 59                                                          POLRT 38
   55 IFIT=1                                                            POLRT 39
      XPR=X                                                             POLRT 40
      YPR=Y                                                             POLRT 41
C        EVALUATE POLYNOMIAL AND DERIVATIVES                            POLRT 42
   59 ICT=0                                                             POLRT 43
   60 UX=0.0                                                            POLRT 44
      UY=0.0                                                            POLRT 45
      V =0.0                                                            POLRT 46
      YT=0.0                                                            POLRT 47
      XT=1.0                                                            POLRT 48
      U=COF(N+1)                                                        POLRT 49
      IF(U) 65,130,65                                                   POLRT 50
   65 DO 70 I=1,N                                                       POLRT 51
      L =N-I+1                                                          POLRT 52
      XT2=X*XT-Y*YT                                                     POLRT 53
      YT2=X*YT+Y*XT                                                     POLRT 54
      U=U+COF(L )*XT2                                                   POLRT 55
      V=V+COF(L )*YT2                                                   POLRT 56
      FI=I                                                              POLRT 57
      UX=UX+FI*XT*COF(L )                                               POLRT 58
      UY=UY-FI*YT*COF(L )                                               POLRT 59
      XT=XT2                                                            POLRT 60
   70 YT=YT2                                                            POLRT 61
      SUMSQ=UX*UX+UY*UY                                                 POLRT 62
      IF(SUMSQ) 75,110,75                                               POLRT 63
   75 DX=(V*UY-U*UX)/SUMSQ                                              POLRT 64
      X=X+DX                                                            POLRT 65
      DY=-(U*UY+V*UX)/SUMSQ                                             POLRT 66
      Y=Y+DY                                                            POLRT 67
   78 IF( ABS(DY)+ ABS(DX)-1.0E-05) 100,80,80                           POLRT 68
C        -STEP ITERATION COUNTER                                        POLRT 69
   80 ICT=ICT+1                                                         POLRT 70
      IF(ICT-500) 60,85,85                                              POLRT 71
   85 IF(IFIT) 100,90,100                                               POLRT 72
   90 IF(IN-5) 50,95,95                                                 POLRT 73
C        SET ERROR CODE TO 3                                            POLRT 74
   95 IER=3                                                             POLRT 75
      GO TO 20                                                          POLRT 76
  100 DO 105 L=1,NXX                                                    POLRT 77
      MT=KJ1-L+1                                                        POLRT 78
      TEMP=XCOF(MT)                                                     POLRT 79
      XCOF(MT)=COF(L)                                                   POLRT 80
  105 COF(L)=TEMP                                                       POLRT 81
      ITEMP=N                                                           POLRT 82
      N=NX                                                              POLRT 83
      NX=ITEMP                                                          POLRT 84
      IF(IFIT) 120,55,120                                               POLRT 85
  110 IF(IFIT) 115,50,115                                               POLRT 86
  115 X=XPR                                                             POLRT 87
      Y=YPR                                                             POLRT 88
  120 IFIT=0                                                            POLRT 89
      IF(X)122,125,122                                                  POLRTM01
  122 IF(ABS(Y)-ABS(X)*1.0E-04)135,125,125                              POLRTM02
  125 ALPHA=X+X                                                         POLRT 91
      SUMSQ=X*X+Y*Y                                                     POLRT 92
      N=N-2                                                             POLRT 93
      GO TO 140                                                         POLRT 94
  130 X=0.0                                                             POLRT 95
      NX=NX-1                                                           POLRT 96
      NXX=NXX-1                                                         POLRT 97
  135 Y=0.0                                                             POLRT 98
      SUMSQ=0.0                                                         POLRT 99
      ALPHA=X                                                           POLRT100
      N=N-1                                                             POLRT101
  140 L1=1                                                              POLRTM03
      L2=2                                                              POLRTM04
      COF(L2)=COF(L2)+ALPHA*COF(L1)                                     POLRTM05
  145 DO 150 L=2,N                                                      POLRT103
  150 COF(L+1)=COF(L+1)+ALPHA*COF(L)-SUMSQ*COF(L-1)                     POLRT104
  155 ROOTI(N2)=Y                                                       POLRT105
      ROOTR(N2)=X                                                       POLRT106
      N2=N2+1                                                           POLRT107
      IF(SUMSQ) 160,165,160                                             POLRT108
  160 Y=-Y                                                              POLRT109
      SUMSQ=0.0                                                         POLRT110
      GO TO 155                                                         POLRT111
  165 IF(N) 20,20,45                                                    POLRT112
      END                                                               POLRT113
```

PADD

Purpose:
Add two polynomials.

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

Description of parameters:
- 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 that vector is not less than the other input vector. The resultant polynomial may have trailing zero coefficients.

Subroutines and function subprograms required:
None.

Method:
Dimension of resultant vector IDIMZ is calculated as the larger of the two input vector dimensions. Corresponding coefficients are then added to form Z.

```
      SUBROUTINE PADD(Z,IDIMZ,X,IDIMX,Y,IDIMY)                  PADD  1
      DIMENSION Z(1),X(1),Y(1)                                  PADD  2
C        TEST DIMENSIONS OF SUMMANDS                            PADD  3
      NDIM=IDIMX                                                PADD  4
      IF (IDIMX-IDIMY) 10,20,20                                 PADD  5
   10 NDIM=IDIMY                                                PADD  6
   20 IF(NDIM) 90,90,30                                         PADD  7
   30 DO 80 I=1,NDIM                                            PADD  8
      IF(I-IDIMX) 40,40,60                                      PADD  9
   40 IF(I-IDIMY) 50,50,70                                      PADD 10
   50 Z(I)=X(I)+Y(I)                                            PADD 11
      GO TO 80                                                  PADD 12
   60 Z(I)=Y(I)                                                 PADD 13
      GO TO 80                                                  PADD 14
   70 Z(I)=X(I)                                                 PADD 15
   80 CONTINUE                                                  PADD 16
   90 IDIMZ=NDIM                                                PADD 17
      RETURN                                                    PADD 18
      END                                                       PADD 19
```

PADDM

Purpose:
Add coefficients of one polynomial to the product of a factor by coefficients of another polynomial.

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

Description of parameters:
- 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 may be in same location as either vector X or vector Y only if the dimension of that vector is not less than the other input vector. The resultant polynomial may have trailing zero coefficients.

Subroutines and function subprograms required:
None.

Method:
Dimension of resultant vector IDIMZ is calculated as the larger of the two input vector dimensions. Coefficient in vector X is then added to coefficient in vector Y multiplied by factor to form Z.

```
      SUBROUTINE PADDM(Z,IDIMZ,X,IDIMX,FACT,Y,IDIMY)            PADDM  1
      DIMENSION Z(1),X(1),Y(1)                                  PADDM  2
C        TEST DIMENSIONS OF SUMMANDS                            PADDM  3
      NDIM=IDIMX                                                PADDM  4
      IF(IDIMX-IDIMY) 10,20,20                                  PADDM  5
   10 NDIM=IDIMY                                                PADDM  6
   20 IF(NDIM) 90,90,30                                         PADDM  7
   30 DO 80 I=1,NDIM                                            PADDM  8
      IF(I-IDIMX) 40,40,60                                      PADDM  9
   40 IF(I-IDIMY) 50,50,70                                      PADDM 10
   50 Z(I)=FACT*Y(I)+X(I)                                       PADDM 11
      GO TO 80                                                  PADDM 12
   60 Z(I)=FACT*Y(I)                                            PADDM 13
      GO TO 80                                                  PADDM 14
   70 Z(I)=X(I)                                                 PADDM 15
   80 CONTINUE                                                  PADDM 16
   90 IDIMZ=NDIM                                                PADDM 17
      RETURN                                                    PADDM 18
      END                                                       PADDM 19
```

PCLA

Purpose:
Move polynomial X to Y.

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

Description of parameters:
- 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.

Remarks:

None.

Subroutines and function subprograms required:

None.

Method:

IDIMY is replaced by IDIMX and vector X is moved to Y.

```
      SUBROUTINE PCLA (Y,IDIMY,X,IDIMX)                          PCLA   1
      DIMENSION X(1),Y(1)                                        PCLA   2
      IDIMY=IDIMX                                                PCLA   3
      IF(IDIMX) 30,30,10                                         PCLA   4
   10 DO 20 I=1,IDIMX                                            PCLA   5
   20 Y(I)=X(I)                                                  PCLA   6
   30 RETURN                                                     PCLA   7
      END                                                        PCLA   8
```

PSUB

Purpose:

Subtract one polynomial from another.

Usage:

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

Description of parameters:

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 that vector is not less than the other input vector. The resultant polynomial may have trailing zero coefficients.

Subroutines and function subprograms required:

None.

Method:

Dimension of resultant vector IDIMZ is calculated as the larger of the two input vector dimensions. Coefficients in vector Y are then subtracted from corresponding coefficients in vector X.

```
      SUBROUTINE PSUB(Z,IDIMZ,X,IDIMX,Y,IDIMY)                   PSUB   1
      DIMENSION Z(1),X(1),Y(1)                                   PSUB   2
C        TEST DIMENSIONS OF SUMMANDS                             PSUB   3
      NDIM=IDIMX                                                 PSUB   4
      IF (IDIMX-IDIMY) 10,20,20                                  PSUB   5
   10 NDIM=IDIMY                                                 PSUB   6
   20 IF (NDIM) 90,90,30                                         PSUB   7
   30 DO 80 I=1,NDIM                                             PSUB   8
      IF (I-IDIMX) 40,40,60                                      PSUB   9
   40 IF (I-IDIMY) 50,50,70                                      PSUB  10
   50 Z(I)=X(I)-Y(I)                                             PSUB  11
      GO TO 80                                                   PSUB  12
   60 Z(I)=-Y(I)                                                 PSUB  13
      GO TO 80                                                   PSUB  14
   70 Z(I)=X(I)                                                  PSUB  15
   80 CONTINUE                                                   PSUB  16
   90 IDIMZ=NDIM                                                 PSUB  17
      RETURN                                                     PSUB  18
      END                                                        PSUB  19
```

PMPY

Purpose:

Multiply two polynomials.

Usage:

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

Description of parameters:

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:

Z cannot be in the same location as X.

Z cannot be in the same location as Y.

Subroutines and function subprograms required:

None.

Method:

Dimension of Z is calculated as IDIMX+IDIMY-1. The coefficients of Z are calculated as sum of products of coefficients of X and Y, whose exponents add up to the corresponding exponent of Z.

```
      SUBROUTINE PMPY(Z,IDIMZ,X,IDIMX,Y,IDIMY)                   PMPY   1
      DIMENSION Z(1),X(1),Y(1)                                   PMPY   2
      IF(IDIMX*IDIMY)10,10,20                                    PMPY   3
   10 IDIMZ=0                                                    PMPY   4
      GO TO 50                                                   PMPY   5
   20 IDIMZ=IDIMX+IDIMY-1                                        PMPY   6
      DO 30 I=1,IDIMZ                                            PMPY   7
   30 Z(I)=0.                                                    PMPY   8
      DO 40 I=1,IDIMX                                            PMPY   9
      DO 40 J=1,IDIMY                                            PMPY  10
      K=I+J-1                                                    PMPY  11
   40 Z(K)=X(I)*Y(J)+Z(K)                                        PMPY  12
   50 RETURN                                                     PMPY  13
      END                                                        PMPY  14
```

PDIV

Purpose:
Divide one polynomial by another.

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

Description of parameters:
- 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 - Error code. 0 is normal, 1 is for zero divisor.

Remarks:
The remainder R replaces X.
The divisor Y remains unchanged.
If dimension of Y exceeds dimension of X, IDIMP is set to zero and calculation is bypassed.

Subroutines and function subprograms required:
PNORM

Method:
Polynomial X is divided by polynomial Y giving integer part P and remainder R such that X = P*Y + R.
Divisor Y and remainder vector get normalized.

```
      SUBROUTINE PDIV(P,IDIMP,X,IDIMX,Y,IDIMY,TOL,IER)                  PDIV   1
      DIMENSION P(1),X(1),Y(1)                                          PDIV   2
      CALL PNORM (Y,IDIMY,TOL)                                          PDIV   3
      IF(IDIMY) 50,50,10                                                PDIV   4
   10 IDIMP=IDIMX-IDIMY+1                                               PDIV   5
      IF(IDIMP) 20,30,60                                                PDIV   6
C        DEGREE OF DIVISOR WAS GREATER THAN DEGREE OF DIVIDEND          PDIV   7
   20 IDIMP=0                                                           PDIV   8
   30 IER=0                                                             PDIV   9
   40 RETURN                                                            PDIV  10
C        Y IS ZERO POLYNOMIAL                                           PDIV  11
   50 IER=1                                                             PDIV  12
      GO TO 40                                                          PDIV  13
C        START REDUCTION                                                PDIV  14
   60 IDIMX=IDIMY-1                                                     PDIV  15
      I=IDIMP                                                           PDIV  16
   70 II=I+IDIMX                                                        PDIV  17
      P(I)=X(II)/Y(IDIMY)                                               PDIV  18
C        SUBTRACT MULTIPLE OF DIVISOR                                   PDIV  19
      DO 80 K=1,IDIMX                                                   PDIV  20
      J=K-1+I                                                           PDIV  21
      X(J)=X(J)-P(I)*Y(K)                                               PDIV  22
   80 CONTINUE                                                          PDIV  23
      I=I-1                                                             PDIV  24
      IF(I) 90,90,70                                                    PDIV  25
C        NORMALIZE REMAINDER POLYNOMIAL                                 PDIV  26
   90 CALL PNORM(X,IDIMX,TOL)                                           PDIV  27
      GO TO 30                                                          PDIV  28
      END                                                               PDIV  29
```

PQSD

Purpose:
Perform quadratic synthetic division.

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

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

Remarks:
None.

Subroutines and function subprograms required:
None.

Method:
The linear remainder A*Z+B.

```
      SUBROUTINE PQSD(A,B,P,Q,X,IDIMX)                                  PQSD   1
      DIMENSION X(1)                                                    PQSD   2
      A=0.                                                              PQSD   3
      B=0.                                                              PQSD   4
      J=IDIMX                                                           PQSD   5
    1 IF(J)3,3,2                                                        PQSD   6
    2 Z=P*A+B                                                           PQSD   7
      B=Q*A+X(J)                                                        PQSD   8
      A=Z                                                               PQSD   9
      J=J-1                                                             PQSD  10
      GO TO 1                                                           PQSD  11
    3 RETURN                                                            PQSD  12
      END                                                               PQSD  13
```

PVAL

Purpose:
Evaluate a polynomial for a given value of the variable.

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

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

Remarks:
None.

Subroutines and function subprograms required:

None.

Method:

Evaluation is done by means of nested multiplication.

```
      SUBROUTINE PVAL(RES,ARG,X,IDIMX)                                  PVAL   1
      DIMENSION X(1)                                                    PVAL   2
      RES=0.                                                            PVAL   3
      J=IDIMX                                                           PVAL   4
    1 IF(J)3,3,2                                                        PVAL   5
    2 RES=RES*ARG+X(J)                                                  PVAL   6
      J=J-1                                                             PVAL   7
      GO TO 1                                                           PVAL   8
    3 RETURN                                                            PVAL   9
      END                                                               PVAL  10
```

PVSUB

Purpose:

Substitute variable of a polynomial by another polynomial.

Usage:

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

Description of parameters:

- 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 polynomial which is substituted for 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).

Remarks:

None.

Subroutines and function subprograms required:

PMPY
PADDM
PCLA

Method:

Variable of polynomial X is substituted by polynomial Y to form polynomial Z. Dimension of new polynomial is (IDIMX-1)*(IDIMY-1)+1. Subroutine requires two work areas.

```
      SUBROUTINE PVSUB(Z,IDIMZ,X,IDIMX,Y,IDIMY,WORK1,WORK2)             PVSUB   1
      DIMENSION Z(1),X(1),Y(1),WORK1(1),WORK2(1)                        PVSUB   2
         TEST OF DIMENSIONS                                             PVSUB   3
      IF (IDIMX-1) 1,3,3                                                PVSUB   4
    1 IDIMZ=0                                                           PVSUB   5
    2 RETURN                                                            PVSUB   6
    3 IDIMZ=1                                                           PVSUB   7
      Z(1)=X(1)                                                         PVSUB   8
      IF (IDIMY*IDIMX-IDIMY) 2,2,4                                      PVSUB   9
    4 IW1=1                                                             PVSUB  10
      WORK1(1)=1.                                                       PVSUB  11
      DO 5 I=2,IDIMX                                                    PVSUB  12
      CALL PMPY(WORK2,IW2,Y,IDIMY,WORK1,IW1)                            PVSUB  13
      CALL PCLA(WORK1,IW1,WORK2,IW2)                                    PVSUB  14
      FACT=X(I)                                                         PVSUB  15
      CALL PADDM(Z,IDIMZ,Z,IDIMZ,FACT,WORK1,IW1)                        PVSUB  16
    5 CONTINUE                                                          PVSUB  17
      GO TO 2                                                           PVSUB  18
      END                                                               PVSUB  19
```

PCLD

Purpose:

Shift of origin (complete linear synthetic division).

Usage:

CALL PCLD(X, IDIMX, U)

Description of parameters:

- 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.

Remarks:

None.

Subroutines and function subprograms required:

None.

Method:

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

```
      SUBROUTINE PCLD (X,IDIMX,U)                                       PCLD   1
      DIMENSION X(1)                                                    PCLD   2
      K=1                                                               PCLD   3
    1 J=IDIMX                                                           PCLD   4
    2 IF (J-K) 4,4,3                                                    PCLD   5
    3 X(J-1)=X(J-1)+U*X(J)                                              PCLD   6
      J=J-1                                                             PCLD   7
      GO TO 2                                                           PCLD   8
    4 K=K+1                                                             PCLD   9
      IF (IDIMX-K) 5,5,1                                                PCLD  10
    5 RETURN                                                            PCLD  11
      END                                                               PCLD  12
```

PILD

Purpose:

Evaluate polynomial and its first derivative for a given argument.

Usage:

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

Description of parameters:

POLY - Value of polynomial.
DVAL - Derivative.
ARGUM - Argument.
X - Vector of coefficients for polynomial, ordered from smallest to largest power.
IDIMX - Dimension of X.

Remarks:

None.

Subroutines and function subprograms required:

PQSD

Method:

Evaluation is done by means of subroutine PQSD (quadratic synthetic division).

```
      SUBROUTINE PILD (POLY,DVAL,ARGUM,X,IDIMX)                          PILD   1
      DIMENSION X(1)                                                     PILD   2
      P=ARGUM+ARGUM                                                      PILD   3
      Q=-ARGUM*ARGUM                                                     PILD   4
      CALL PQSD (DVAL,POLY,P,Q,X,IDIMX)                                  PILD   5
      POLY=ARGUM*DVAL+POLY                                               PILD   6
      RETURN                                                             PILD   7
      END                                                                PILD   8
```

PDER

Purpose:

Find derivative of a polynomial.

Usage:

CALL PDER(Y, IDIMY, X, IDIMX)

Description of parameters:

Y - Vector of coefficients for derivative, 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.

Remarks:

None.

Subroutines and function subprograms required:

None.

Method:

Dimension of Y is set at dimension of X less one. Derivative is then calculated by multiplying coefficients by their respective exponents.

```
      SUBROUTINE PDER(Y,IDIMY,X,IDIMX)                                   PDER   1
      DIMENSION X(1),Y(1)                                                PDER   2
C        TEST OF DIMENSION                                               PDER   3
      IF (IDIMX-1) 3,3,1                                                 PDER   4
    1 IDIMY=IDIMX-1                                                      PDER   5
      EXPT=0.                                                            PDER   6
      DO 2 I=1,IDIMY                                                     PDER   7
      EXPT=EXPT+1.                                                       PDER   8
    2 Y(I)=X(I+1)*EXPT                                                   PDER   9
      GO TO 4                                                            PDER  10
    3 IDIMY=0                                                            PDER  11
    4 RETURN                                                             PDER  12
      END                                                                PDER  13
```

PINT

Purpose:

Find integral of a polynomial with constant of integration equal to zero.

Usage:

CALL PINT(Y, IDIMY, X, IDIMX)

Description of parameters:

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.

Remarks:

None.

Subroutines and function subprograms required:

None.

Method:

Dimension of Y is set at dimension of X plus one, and the constant term is set to zero. Integral is then calculated by dividing coefficients by their respective exponents.

```
      SUBROUTINE PINT(Y,IDIMY,X,IDIMX)                                   PINT   1
      DIMENSION X(1),Y(1)                                                PINT   2
      IDIMY=IDIMX+1                                                      PINT   3
      Y(1)=0.                                                            PINT   4
      IF(IDIMX)1,1,2                                                     PINT   5
    1 RETURN                                                             PINT   6
    2 EXPT=1.                                                            PINT   7
      DO 3 I=2,IDIMY                                                     PINT   8
      Y(I)=X(I-1)/EXPT                                                   PINT   9
    3 EXPT=EXPT+1.                                                       PINT  10
      GO TO 1                                                            PINT  11
      END                                                                PINT  12
```

PGCD

Purpose:

Determine greatest common divisor of two polynomials.

Usage:

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

Description of parameters:

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 where:

IER=0 No error.

IER=1 X or Y is zero polynomial.

Remarks:

IDIMX must be greater than IDIMY.

IDIMY=1 on return means X and Y are prime, the GCD is a constant.

Subroutines and function subprograms required:

PDIV

PNORM

Method:

Greatest common divisor of two polynomials X and Y is determined by means of Euclidean algorithm. Coefficient vectors X and Y are destroyed and greatest common divisor is generated in Y.

```
      SUBROUTINE PGCD(X,IDIMX,Y,IDIMY,WORK,EPS,IER)                    PGCD   1
      DIMENSION X(1),Y(1),WORK(1)                                      PGCD   2
C        DIMENSION REQUIRED FOR VECTOR NAMED  WORK  IS   IDIMX-IDIMY+1 PGCD   3
    1 CALL PDIV(WORK,NDIM,X,IDIMX,Y,IDIMY,EPS,IER)                     PGCD   4
      IF(IER) 5,2,5                                                    PGCD   5
    2 IF(IDIMX) 5,5,3                                                  PGCD   6
C        INTERCHANGE X AND Y                                           PGCD   7
    3 DO 4 J=1,IDIMY                                                   PGCD   8
      WORK(J)=X(J)                                                     PGCD   9
      X(J)=Y(J)                                                        PGCD  10
    4 Y(J)=WORK(J)                                                     PGCD  11
      NDIM=IDIMX                                                       PGCD  12
      IDIMX=IDIMY                                                      PGCD  13
      IDIMY=NDIM                                                       PGCD  14
      GO TO 1                                                          PGCD  15
    5 RETURN                                                           PGCD  16
      END                                                              PGCD  17
```

PNORM

Purpose:

Normalize coefficient vector of a polynomial.

Usage:

CALL PNORM(X, IDIMX, EPS)

Description of parameters:

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, result is a zero polynomial with IDIMX=0 but vector X remains intact.

Subroutines and function subprograms required:

None.

Method:

Dimension of vector X is reduced by one for each trailing coefficient with an absolute value less than or equal to EPS.

```
      SUBROUTINE PNORM(X,IDIMX,EPS)                                    PNORM  1
      DIMENSION X(1)                                                   PNORM  2
    1 IF(IDIMX) 4,4,2                                                  PNORM  3
    2 IF(ABS(X(IDIMX))-EPS) 3,3,4                                      PNORM  4
    3 IDIMX=IDIMX-1                                                    PNORM  5
      GO TO 1                                                          PNORM  6
    4 RETURN                                                           PNORM  7
      END                                                              PNORM  8
```

APPENDIX A: STORAGE REQUIREMENTS

The following table lists the number of characters of storage required by each of the subroutines in the Scientific Subroutine Package. The figures given were obtained by using 1130 Monitor FORTRAN, Version 2, Modification Level 1. Also noted are the subroutines or function subprograms that are called or are used by a given subroutine.

Name	Function	Storage Required (Words)
STATISTICS		
Data Screening		
TALLY	totals, means, standard deviations, minimums, maximums	356
BOUND	selection of observations within bounds	206
SUBST	subset selection from observation matrix (needs user's Boolean subroutine)	224
ABSNT	detection of missing data	94
TAB1	tabulation of data (1 variable)	612
TAB2	tabulation of data (2 variables)	1140
SUBMX	build subset matrix	134
Elementary Statistics		
MOMEN	first four moments	404
TTSTT	tests on population means	538
Correlation		
CORRE	means, standard deviations, and correlations (needs user's subroutine to get data)	1164
Multiple Linear Regression		
(Usually requires CORRE, ORDER, MINV, MULTR in sequence)		
ORDER	rearrangement of intercorrelations	206
MULTR	multiple regression and correlation	518
Polynomial Regression		
(Usually requires GDATA, ORDER, MINV, MULTR in sequence)		
GDATA	data generation	668
Canonical Correlation		
(Usually requires CORRE, CANOR, MINV, NROOT, EIGEN in sequence)		
CANOR	canonical correlation (CANOR calls MINV and NROOT)	1132
NROOT	eigenvalues and eigenvectors of a special nonsymmetric matrix (NROOT calls EIGEN)	752
Analysis of Variance		
(Usually requires AVDAT, AVCAL, MEANQ in sequence)		
AVDAT	data storage allocation	326
AVCAL	Σ and Δ operation	268
MEANQ	mean square operation	560

Name	Function	Storage Required (Words)
Discriminant Analysis		
(Usually requires DMATX, MINV, DISCR in sequence)		
DMATX	means and dispersion matrix	422
DISCR	discriminant functions	980
Factor Analysis		
(Usually requires CORRE, EIGEN, TRACE, LOAD, VARMX in sequence)		
TRACE	cumulative percentage of eigenvalues	160
LOAD	factor loading	98
VARMX	varimax rotation	1186
Time Series		
AUTO	autocovariances	180
CROSS	crosscovariances	248
SMO	application of filter coefficients (weights)	166
EXSMO	triple exponential smoothing	274
Nonparametric Statistics		
CHISQ	x^2 test for a contingency table	490
UTEST	Mann-Whitney U-test (UTEST calls RANK and TIE)	242
TWOAV	Friedman two-way analysis of variance (TWOAV calls RANK)	324
QTEST	Cochran Q-test	232
SRANK	Spearman rank correlation (SRANK calls RANK and TIE)	378
KRANK	Kendall rank correlation (KRANK calls RANK and TIE)	524
WTEST	Kendall coefficient of concordance (WTEST calls RANK and TIE)	498
RANK	rank observations	216
TIE	calculation of ties in ranked observations	196
Random Number Generators		
RANDU	uniform random numbers	52
GAUSS	normal random numbers (GAUSS calls RANDU)	68
MATHEMATICS		
Special Matrix Operations		
MINV	matrix inversion	784
EIGEN	eigenvalues and eigenvectors of a real, symmetric matrix	1058
Matrices		
SIMQ	solution of simultaneous linear, algebraic equations	540
GMADD	add two general matrices	52
GMSUB	subtract two general matrices	52
GMPRD	product of two general matrices	156
GMTRA	transpose of a general matrix	88
GTPRD	transpose product of two general matrices	152
MADD	add two matrices (calls LOC)	226

Name	Function	Storage Required (Words)
MSUB	subtract two matrices (calls LOC)	226
MPRD	matrix product (row into column) (calls LOC)	230
MTRA	transpose a matrix (calls MCPY)	108
TPRD	transpose product (calls LOC)	230
MATA	transpose product of matrix by itself (calls LOC)	194
SADD	add scalar to matrix (calls LOC)	56
SSUB	subtract scalar from a matrix (calls LOC)	56
SMPY	matrix multiplied by a scalar (calls LOC)	56
SDIV	matrix divided by a scalar (calls LOC)	66
RADD	add row of one matrix to row of another matrix (calls LOC)	90
CADD	add column of one matrix to column of another matrix (calls LOC)	92
SRMA	scalar multiply row and add to another row	108
SCMA	scalar multiply column and add to another column	110
RINT	interchange two rows	94
CINT	interchange two columns	96
RSUM	sum the rows of a matrix (calls LOC)	98
CSUM	sum the columns of a matrix (calls LOC)	98

Name	Function	Storage Required (Words)
RTAB	tabulate the rows of a matrix (calls LOC, RADD)	198
CTAB	tabulate the columns of a matrix (calls LOC, CADD)	198
RSRT	sort matrix rows (calls LOC)	308
CSRT	sort matrix columns (calls LOC and CCPY)	306
RCUT	partition row-wise (calls LOC)	162
CCUT	partition column-wise (calls LOC)	162
RTIE	adjoin two matrices row-wise (calls LOC)	178
CTIE	adjoin two matrices column-wise (calls LOC)	166
MCPY	matrix copy (calls LOC)	52
XCPY	copy submatrix from given matrix (calls LOC)	128
RCPY	copy row of matrix into vector (calls LOC)	78
CCPY	copy column of matrix into vector (calls LOC)	78
DCPY	copy diagonal of matrix into vector (calls LOC)	58
SCLA	matrix clear and add scalar (calls LOC)	52
DCLA	replace diagonal with scalar (calls LOC)	50
MSTR	storage conversion (calls LOC)	116
MFUN	matrix transformation by a function	66
RECP	reciprocal function for MFUN	44

Name	Function	Storage Required (Words)
LOC	location in compressed-stored matrix	108
ARRAY	vector storage--double dimensioned conversion	198
Integration and Differentiation		
QSF	integral of tabulated function	806
QATR	integral of given function by trapezoidal rule.	386
Ordinary Differential Equations		
RK1	integral of first-order differential equation by Runge-Kutta method (needs user function subprogram)	468
RK2	tabulated integral of first-order differential equation by Runge-Kutta method (needs user function subprogram)	210
RKGS	solution of a system of first-order differential equations by Runge-Kutta method (uses given initial values)	1174
Fourier Analysis		
FORIF	Fourier analysis of a given function (needs user function subprogram)	292
FORIT	Fourier analysis of a tabulated function	284
Special Operations and Mathematical Functions		
GAMMA	gamma function	260
LEP	Legendre polynomial	132
BESJ	J Bessel function	448
BESY	Y Bessel function	704
BESI	I Bessel function	414
BESK	K Bessel function	844
CEL1	elliptic integral of the first kind	126
CEL2	elliptic integral of the second kind	200
EXPI	exponential integral	262
SICI	sine cosine integral	366
CS	Fresnel integrals	310
Roots of Nonlinear Functions		
RTWI	refine estimate of root of Wegstein's iteration (needs user function subprogram)	208
RTMI	determine root within a range by Mueller's iteration (needs user function subprogram)	536
RTNI	refine estimate of root by Newton's iteration (needs user function subprogram)	172
Roots of Polynomial		
POLRT	real and complex roots of polynomial	820
Polynomial Operations		
PADD	add two polynomials	110
PADDM	multiply polynomial by constant and add to another polynomial	118
PCLA	replace one polynomial by another	48

Name	Function	Storage Required (Words)
PSUB	subtract one polynomial from another	112
PMPY	multiply two polynomials	142
PDIV	divide one polynomial by another (calls PNORM)	198
PQSD	quadratic synthetic division of a polynomial	78
PVAL	value of a polynomial	54
PVSUB	substitute variable of polynomial by another polynomial (calls PMPY, PADDM, PCLA)	132
PCLD	complete linear division	74
PILD	evaluate polynomial and its first derivative (calls PQSD)	56
PDER	derivative of a polynomial	88
PINT	integral of a polynomial	88
PGCD	greatest common divisor of two polynomials (calls PDIV and PNORM)	108
PNORM	normalize coefficient vector of polynomial	48

APPENDIX B: ACCURACY OF SUBROUTINES

The subroutines in SSP can be broken down into three major categories from the standpoint of accuracy. They are: subroutines having little or no affect on accuracy; subroutines whose accuracy is dependent on the characteristics of the input data; and subroutines in which definite statements on accuracy can be made.

SUBROUTINES HAVING LITTLE OR NO AFFECT ON ACCURACY

The following subroutines do not materially affect the accuracy of the results, either because of the simple nature of the computation or because they do not modify the data:

TALLY	totals, means, standard deviations, minimums, and maximums
BOUND	selection of observations within bounds
SUBST	subset selection from observation matrix
ABSNT	detection of missing data
TAB1	tabulation of data (1 variable)
TAB2	tabulation of data (2 variables)
SUBMX	build subset matrix
MOMEN	first four moments
TTSTT	tests on population means
ORDER	rearrangement of intercorrelations
AVDAT	data storage allocation
TRACE	cumulative percentage of eigenvalues
CHISQ	χ^2 test for a contingency table
UTEST	Mann-Whitney U-test
TWOAV	Friedman two-way analysis of variance
QTEST	Cochran Q-test
SRANK	Spearman rank correlation
KRANK	Kendall rank correlation
WTEST	Kendall coefficient of concordance
RANK	rank observations
TIE	calculation of ties in ranked observations
RANDU	uniform random numbers
GAUSS	normal random numbers
GMADD	add two general matrices
GMSUB	subtract two general matrices
GMPRD	product of two general matrices
GMTRA	transpose of a general matrix
GTPRD	transpose product of two general matrices
MADD	add two matrices
MSUB	subtract two matrices
MPRD	matrix product (row into column)
MTRA	transpose a matrix
TPRD	transpose a product
MATA	transpose product of matrix by itself
SADD	add scalar to matrix
SSUB	subtract scalar from a matrix
SMPY	matrix multiplied by a scalar
SDIV	matrix divided by a scalar
RADD	add row of one matrix to row of another matrix
CADD	add column of one matrix to column of another matrix
SRMA	scalar multiply row and add to another row
SCMA	scalar multiply column and add to another column
RINT	interchange two rows
CINT	interchange two columns
RSUM	sum the rows of a matrix
CSUM	sum the columns of a matrix
RTAB	tabulate the rows of a matrix
CTAB	tabulate the columns of a matrix
RSRT	sort matrix rows
CSRT	sort matrix columns
RCUT	partition row-wise
CCUT	partition column-wise
RTIE	adjoin two matrices row-wise
CTIE	adjoin two matrices column-wise
MCPY	matrix copy
XCPY	copy submatrix from given matrix
RCPY	copy row of matrix into vector
CCPY	copy column of matrix into vector
DCPY	copy diagonal of matrix into vector
SCLA	matrix clear and add scalar
DCLA	replace diagonal with scalar

MSTR	storage conversion
MFUN	matrix transformation by a function
RECP	reciprocal function for MFUN
LOC	location in compressed-stored matrix
CONVT	single precision, double precision conversion
ARRAY	vector storage--double dimensioned conversion
PADD	add two polynomials
PADDM	multiply polynomial by constant and add to another polynomial
PCLA	replace one polynomial by another
PSUB	subtract one polynomial from another
PMPY	multiply two polynomials
PDIV	divide one polynomial by another
PQSD	quadratic synthetic division of a polynomial
PVAL	value of a polynomial
PVSUB	substitute variable of polynomial by another polynomial
PCLD	complete linear division
PILD	evaluate polynomial and its first derivative
PDER	derivative of a polynomial
PINT	integral of a polynomial
PGCD	greatest common divisor of two polynomials
PNORM	normalize coefficient vector of polynomial

SUBROUTINES WHOSE ACCURACY IS DATA DEPENDENT

The accuracy of the following subroutines cannot be predicted because it is dependent on the characteristics of the input data and on the size of the problem. The programmer using these subroutines must be aware of the limitations dictated by numerical analyses considerations. It cannot be assumed that the results are accurate simply because subroutine execution is completed. Subroutines in this category are:

CORRE	means, standard deviations, and correlations
MULTR	multiple regression and correlation
GDATA	data generation
CANOR	canonical correlation
NROOT	eigenvalues and eigenvectors of a special nonsymmetric matrix
AVCAL	Σ and Δ operation
MEANQ	mean square operation
DMATX	means and dispersion matrix
DISCR	discriminant functions
LOAD	factor loading
VARMX	varimax rotation
AUTO	autocovariances
CROSS	crosscovariances
SMO	application of filter coefficients (weights)
EXSMO	triple exponential smoothing
MINV	matrix inversion
EIGEN	eigenvalues and eigenvectors of a real, symmetric matrix
SIMQ	solution of simultaneous linear, algebraic equations
QSF	integral of tabulated function by Simpson's Rule
QATR	integral of given function by trapezoidal rule
RK1	integral of first-order differential equation by Runge-Kutta method
RK2	tabulated integral of first-order differential equation by Runge-Kutta method
RKGS	solution of a system of first-order differential equations by Runge-Kutta method
FORIF	Fourier analysis of a given function
FORIT	Fourier analysis of a tabulated function
RTWI	refine estimate of root by Wegstein's iteration
RTMI	determine root within a range by Mueller's iteration
RTNI	refine estimate of root by Newton's iteration
POLRT	real and complex roots of polynomial

SUBROUTINES WITH DEFINITE ACCURACY CHARACTERISTICS

This table was developed by comparing floating-point results from the subroutines with the tables given in Abramowitz and Stegun*. In certain cases the reference table gave results in fixed-point form. In these cases the maximum differences below are given in terms of number of decimal places (d.p.) which agreed, rather than number of significant digits (s.d.) which agree. In compiling maximum differences, the maximum was taken over the set of points indicated in the table. The average difference was normally much smaller.

The notation x = a (b) c implies that a, a + b, a + 2b, ..., c were the arguments (x) used.

Name	Functions	Remarks	Allowable Parameter Range	Range Checked with references*	Maximum Difference s.d.=significant digits d.p.=decimal places
GAMMA	$\Gamma(x)$ (gamma)		$x \leq 34.5$, and x not within 10^{-6} of zero or a negative integer	x = .1 (.1) 3	2 in 6th s.d.
				x = 1 (1) 34	1 in 6th s.d.
LEP	$P_n(x)$ (Legendre)		$-1 \leq x \leq 1$	x = 0 (.2) 1 n = 2, 3	3 in 6th s.d.
			$n \geq 0$	n = 9, 10	1 in 5th s.d.
BESJ	$J_n(x)$ (Bessel)	(The accuracy Factor, D, used in the program was 10^{-5}.)	$x > 0$; $n > 0$	x = 1 (1) 17 n = 0, 1, 2	8 in 6th s.d.
			when $x \leq 15$; $n < 20 + 10x^{-x^{2/3}}$	n = 3 (1) 9 x = 1 (1) n-2	1 in 5th s.d.
			when $x > 15$, $n < 90 + x/2$	n = 3 (1) 9 x = n - 1 (1) 20	1 in 5th d.p.
				x = 1, 2, 5, 10, 50 n = 10 (10) 50**	3 in 6th s.d.
BESY	$Y_n(x)$ (Bessel)		$n \geq 0$ $x > 0$	x = 1 (1) 17 n = 0, 1, 2	9 in 6th s.d.
				n = 3 (1) 9 x = 1 (1) n-2	1 in 5th s.d.
				n = 3 (1) 9 x = n-1 (1) 20	1 in 5th d.p.
				x = 1, 2, 5, 10, 50 n = 10 (10) 50**	3 in 5th s.d.
BESI	$I_n(x)$ (Bessel)	(Table values are $e^{-x}I_n(x)$. maximum difference is for these values)	$x > 0$ $0 \leq n \leq 30$	x = 1 (1) 20 n = 0, 1	8 in 7th s.d.
				x = 5 (1) 20 n = 2	6 in 7th s.d.
				x = 1 (1) 20 n = 3 (1) 9	1 in 5th s.d.
		(Table values are $I_n(x)$)		x = 1, 2, 5, 10 n = 10, 20, 30**	8 in 7th s.d.

Subroutines with Definite Accuracy Characteristics (continued)

Name	Functions	Remarks	Allowable Parameter Range	Range Checked with references*	Maximum Difference s.d.=significant digits d.p.=decimal places
BESK	$K_n(x)$ (Bessel)	(Table values are $e^x K_n(x)$. These were used for maximum differences)	$x > 0$ $n \geq 0$	$x = 1\ (1)\ 20$ $n = 0, 1$	8 in 7th s.d.
				$x = 5\ (1)\ 20$ $n = 2$	9 in 7th s.d.
				$x = 1\ (1)\ 20$ $n = 3\ (1)\ 9$	1 in 5th s.d.
		(Tabled values are $K_n(x)$		$x = 1, 2, 5, 10, 50$ $n = 10\ (10)\ 50$**	1 in 6th s.d.
CEL1	K (k) (elliptic 1st integral)	(Tabled values are K(m); $m = k^2$	$-1 \leq k \leq 1$	$m = 0\ (.1)\ .9$	1 in 7th s.d.
CEL2	(Generalized Integral of 2nd kind)	K(m) when $A = B = 1$ E(m) when $A = 1$, $B = ck^2$ where $m = k^2$	$-1 \leq k \leq 1$	$m = 0\ (.1)\ .9$	1 in 7th s.d.
				$m = 0\ (.1)\ .9$	1 in 7th s.d.
EXPI	Exponential Integral	$-Ei(-x)$ when $X < 0$	$x \geq -4$	$x = -.5\ (.-5)\ -2$	0 in 7th s.d.
				$x = -2.5\ (-.5)\ -4$	1 in 7th s.d.***
		$E_1(x)$ when $x > 0$		$x = .5\ (.5)\ 2$	2 in 7th s.d.
				$x = 2.5\ (.5)\ 4$	6 in 5th s.d.***
				$x = 4.5\ (.5)\ 8$	3 in 7th s.d.***
SICI	$s_i(x)$ (sine integral)		none	$x = 1\ (1)\ 10$ $x = 10\pi$	3 in 7th s.d. 0 in 7th s.d.
SICI	C i (x) (cosine integral)		none	$x = 1\ (1)\ 10$ $x = 10\pi$	3 in 7th s.d. 0 in 5th s.d.
CS	$C_2(u)$ (Fresnel) $\mu = \frac{1}{2}\pi x^2$		none	$x = .1, .3, .6, .8$	1 in 6th s.d.
				$x = 1\ (1)\ 5$	2 in 7th s.d.
CS	$S_2(u)$ (Fresnel) $\mu = \frac{1}{2}\pi x^2$		none	$x = .1, .3, .6, .8$	1 in 4th s.d.
				$x = 1\ (1)\ 5$	3 in 7th s.d.

*Handbook of Mathematical Functions, Abramowitz and Stegun, National Bureau of Standards publication.

**Results outside the range of the 1130 are set to zero or machine infinity. Results are subject to compatability of x and n.

***Tabled results, used for maximum difference, were given for $xe^x E_i(-x)$ and $xe^x E_1(x)$

APPENDIX C: TIMING

1. Sample program SOLN was chosen to exemplify the overall timing of a problem. In all cases the 1442 Card Reader, Model 7, is used for input and all necessary subroutines are already on disk. (Core speed: 3.6 μs.)

   a. Compile time, using a LIST ALL card (gives a program listing of its 56 cards and a memory map which includes variable allocations, statement allocations, features supported, called subprograms, integer constants, and core requirements), requires 1 minute 32 seconds on the 1132 Printer. (Compile time, minus the LIST ALL card, requires 36 seconds.)

   b. To store the program on disk takes 10 seconds.

   c. After the XEQ control card is read, the computer uses 17 seconds to locate the necessary subprograms and the main program, and to load them in core.

   d. Execution time is four seconds. Output printing time is 53 seconds on an 1132 Printer and 3 minutes 32 seconds on the console typewriter.

2. To illustrate the computational time used by an IBM 1130 computer, the following program was selected:

```
      DIMENSION A(1600),L(40),M(40)
      IX=3
    2 PAUSE 1
      DO 1 I=1,1600
      CALL RANDU (IX,IY,Y)
      IX=IY
    1 A(I)=Y
      PAUSE 2
      CALL MINV (A,10,D,L,M)
      PAUSE 3
      CALL MINV (A,15,D,L,M)
      PAUSE 4
      CALL MINV (A,20,D,L,M)
      PAUSE 5
      CALL MINV (A,30,D,L,M)
      PAUSE 6
      CALL MINV (A,40,D,L,M)
      PAUSE 7
      GO TO 2
      END
```

   a. RANDU - random number generator subroutine. To generate 1600 numbers, using subroutine RANDU, execution time is 5 seconds.

   b. MINV - matrix inversion subroutine. Matrix inversion, using subroutine MINV, is performed on five different sized matrices, with the following results in execution time:

      (1) The 10 x 10 matrix uses 4 seconds.
      (2) The 15 x 15 matrix uses 12 seconds.
      (3) The 20 x 20 matrix uses 27 seconds.
      (4) The 30 x 30 matrix uses 1 minute 28 seconds.
      (5) The 40 x 40 matrix uses 3 minutes 27 seconds.

SAMPLE PROBLEM TIMING

The table below gives sample problem times from the reading of the XEQ card to the printing, on the 1132 Printer, of the last output line:

Problem	Time
DASCR	2 min. 20 sec. (5 min. 30 sec. using the console typewriter)
ADSAM	1 min. 25 sec.
ANOVA	55 sec.
EXPON	1 min. 5 sec.
FACTO	1 min. 55 sec.
MCANO	1 min. 55 sec.
MDISC	2 min. 12 sec.
POLRG	2 min. 53 sec.
QDINT	30 sec.
REGRE	2 min. 25 sec.
RKINT	55 sec.
SMPRT	30 sec.
SOLN	1 min. 15 sec.

APPENDIX D: SAMPLE PROGRAMS

This appendix describes a set of sample programs designed to illustrate typical applications of the scientific subroutines. The sample programs also make use of certain user-written special sample subroutines. Such subroutines are, of course, to be taken only as typical solutions to the problem under consideration, each user being urged to tailor such subroutines to his own specific requirements.

A "Guide to the Sample Programs" immediately follows this introduction. The guide indicates the location of the sample program (if any) calling a particular subroutine of the SSP or referencing a special sample subroutine. The SSP listings are not repeated in this appendix; to locate such listings refer to "Guide to Subroutines" in the introduction.

Listings of the special sample subroutines (HIST, MATIN, PLOT, MXOUT, BOOL, DATA, and FUN) are provided immediately following each sample program. The subroutines DATA, MATIN, and MXOUT are used with several sample programs, and for purposes of clarity the listings of these special user-written routines are repeated with each sample program.

GUIDE TO THE SAMPLE PROGRAMS

SAMPLE PROGRAM DESCRIPTION

The specific requirements for each sample program, including problem description, subroutines, program capacity, input, output, operating instructions, error messages, program modifications, and timing, as well as listings of data inputs and program results, are given in the documentations of the individual sample programs.

There are, however, several significant facts, which apply to all these sample programs.

1. Data input to programs produced by 1130 FORTRAN is required to be right justified within a field, even if the data includes decimal points. Only leading blanks are permitted.

2. All sample programs as distributed will run on an 8K Model IIB with 1132 Printer and 1442 Card Read Punch, Model 6 or 7. If the user *has different*

card I/O devices, he must change the *IOCS card and the first READ instruction of each sample program to conform to his configuration.

3. All of the output format statements in the sample main programs and the sample subroutines specify the console typewriter as the output device. However, the logical unit numbers for input and output are optional. The first card of the sample problem data deck defines the input/output units for a specific run, and is read from the principal card reader by the sample main program. Format for this card is as follows:

Column 2 contains the logical unit number for output

Column 4 contains the logical unit number for input

4. The IOCS card, included with each sample main program, specifies three devices (CARD, TYPEWRITER, 1132 PRINTER). The user should include only those I/O devices employed by the program, thus eliminating any unnecessary Monitor subroutines.

5. Since core storage for the IBM 1130 Model II B computer is 8K, only a limited number of the sample programs have ample storage area for increases in dimension statements. The majority of the programs are now dimensioned so near maximum storage size that any increases in the dimension would create system overlays (SOCAL's) or would necessitate the use of a LOCAL overlay area.

6. For each sample program given below, there is a schematic diagram showing deck setup. This schematic gives a general description of deck requirements. Specific details pertaining to three different situations should be understood. To follow the discussion of the three cases for all sample programs, consider Figure 10.

a. Initial run of a sample program under the disk monitor system: All required monitor control cards are distributed with decks. If the deck setup given in Figure 10 is used, the final card of the routine DASCR, the //XEQ card (which is a monitor control card), should be taken out of the routine DASCR and placed after the *STORE card which has stored the routine LOC on the disk. With this change, DASCR will be compiled, stored on disk (with all of its required routines), and then will execute. After this initial run is complete, the second case can be considered (b, below).

b. After the initial run of a sample program under the disk monitor system, following runs can be made by using only the //XEQ card and any required *LOCAL cards, followed by data. This case assumes that all routines are on the disk.

c. Running sample programs under Card FORTRAN (1130-FO-001) (non-disk system): All monitor control cards (see the Application Directory) must now be removed from decks. Using Figure 10, consider that the labeled decks refer to object programs which were previously compiled using Card FORTRAN (C26-3629). With this consideration, noting the binary loaders and library required as stated under "Object Deck Loading Procedures" in the 1130 Card/Paper Tape Programming System Operator's Guide, and with decks in Figure 10 order, DASCR will run.

NOTE: Remarks in (a) above about changes in placement of //XEQ cards pertain also to any required *LOCAL cards, which must succeed the //XEQ cards.

A fourth situation may also be considered. If the user has all subroutines stored on the disk, and none of the sample problems are on the disk, then any individual sample problem will run as it was distributed in card form.

A LOCAL card, following the XEQ Monitor control card, allows the user to designate all subroutines to be loaded into a LOCAL overlay area on call at execution time. For the function of SOCAL and the use of LOCAL, the reader is referred to IBM 1130 Disk Monitor System, Version 2, Programming and Operator's Guide (C26-3717). The sample programs employ the LOCAL facility.

DATA SCREENING

Problem Description

A set of observations is read along with information on propositions to be satisfied and limits on a selected variable. From this input a subset is obtained and a histogram of frequency over given class intervals is plotted for the selected variable. Total, average, standard deviation, minimum, and maximum are calculated for the selected variable. This procedure is repeated until all sets of input data have been processed.

Program

Description

The data screening sample program consists of a main routine, DASCR, and six subroutines:

SUBST
TAB1
LOC

} are from the Scientific Subroutine Package

MATIN	is a sample input routine
HIST	is a sample program for plotting a histogram
BOOL	refer to subroutine SUBST

Capacity

The maximum size of matrix of observations has been set at 1000 elements, the number of observations at 200, and the number of conditions at 21. Therefore, if a problem satisfies the above conditions, no modification to the sample program is necessary. However, if the maximum sizes must be increased, the dimension statements in the sample main program must be modified to handle this particular problem. The general rules for program modification are described later.

Input

One I/O Specification card defines input/output units (see "Sample Program Descriptions".)

A parameter card with the following format must precede each matrix of observations:

Columns	Contents	For Sample Problem
1 - 2	Blank	
3 - 6	Up to four digit identification code (numeric only)	0001
7 - 10	Number of observations	0100
11 - 14	Number of variables	0004

Matrix of Observations

Each matrix of observations must be followed by a card with a 9 punch in column 1.

The condition matrix and bounds data are preceded by a parameter card containing the number of conditions and the variable to be selected for analysis:

Columns	Contents	For Sample Problem
1 - 2	Number of conditions	02
3 - 4	Variable to be selected	03

UBO Vector

A card with an asterisk in column 1 must follow the UBO vector. A blank card after the last set of input data terminates the run.

Data Cards

1. The observation matrix: Data cards have seven fields of ten columns each, starting in column one. The decimal point may appear anywhere in a field or may be omitted, if the number is an integer. However, all numbers must be right justified even if the decimal point is punched. The number in each field may be preceded by blanks. All values for an observation are punched consecutively and may continue from card to card. However, a new observation must start in the first field of the next card.
2. The condition matrix (see description in the subroutine SUBST): Each ten-column field contains a condition to be satisfied. The first two columns contain the variable number (right justified), the third column the relational code, and the last seven columns of each field a floating-point number. There may be as many as seven conditions per card and a total of three cards or 21 conditions.
3. The UBO vector (see description in the subroutine TAB1): The UBO vector is punched in three fields of ten columns each as a floating-point number.

Deck Setup

The deck setup is shown in Figure 10.

Sample

A listing of input cards for the sample problem is presented at the end of the sample main program.

Output

Description

The output consists of the subset vector showing which observations are rejected (zero) and accepted (nonzero), summary statistics for the selected variable, and a histogram of frequencies versus intervals for that variable.

Sample

The output listing for the sample problem is shown in Figure 11.

Program Modification

Noting that storage problems may result, as previously discussed in "Sample Program Description", program capacity can be increased or decreased by making changes to the DIMENSION statement. In order to familiarize the user with the program modification, the following general rules are supplied in terms of the sample problem:

1. Changes in the dimension statement of the main program, DASCR.
   a. The dimension of array A must be greater than or equal to the number of elements in the observation matrix. For the sample problems the value is 400.

b. The dimension of array C must be greater than or equal to the number of conditions, c times 3. For the sample problem this product is 6 = 2 x 3.

c. The dimension of array S must be greater than or equal to the number of observations, m. Since there are 100 observations in the sample problem the value of m is 100.

d. The dimension of array R must be greater than or equal to the number of conditions, c. For the sample problem the value of c is 2.

e. The dimensions of array FREQ and PCT must be greater than or equal to the number of intervals for the selected variable. For the sample problem this value is 20.

2. Insert the dimension size for A in the third argument of the CALL MATIN statement (following statement 24).

3. Subroutine BOOL can be replaced if the user wishes to use a different boolean expression (see description in subroutine SUBST). The boolean expression provided in the sample program is for both conditions to be satisfied:

$$T = R(1) * R(2)$$

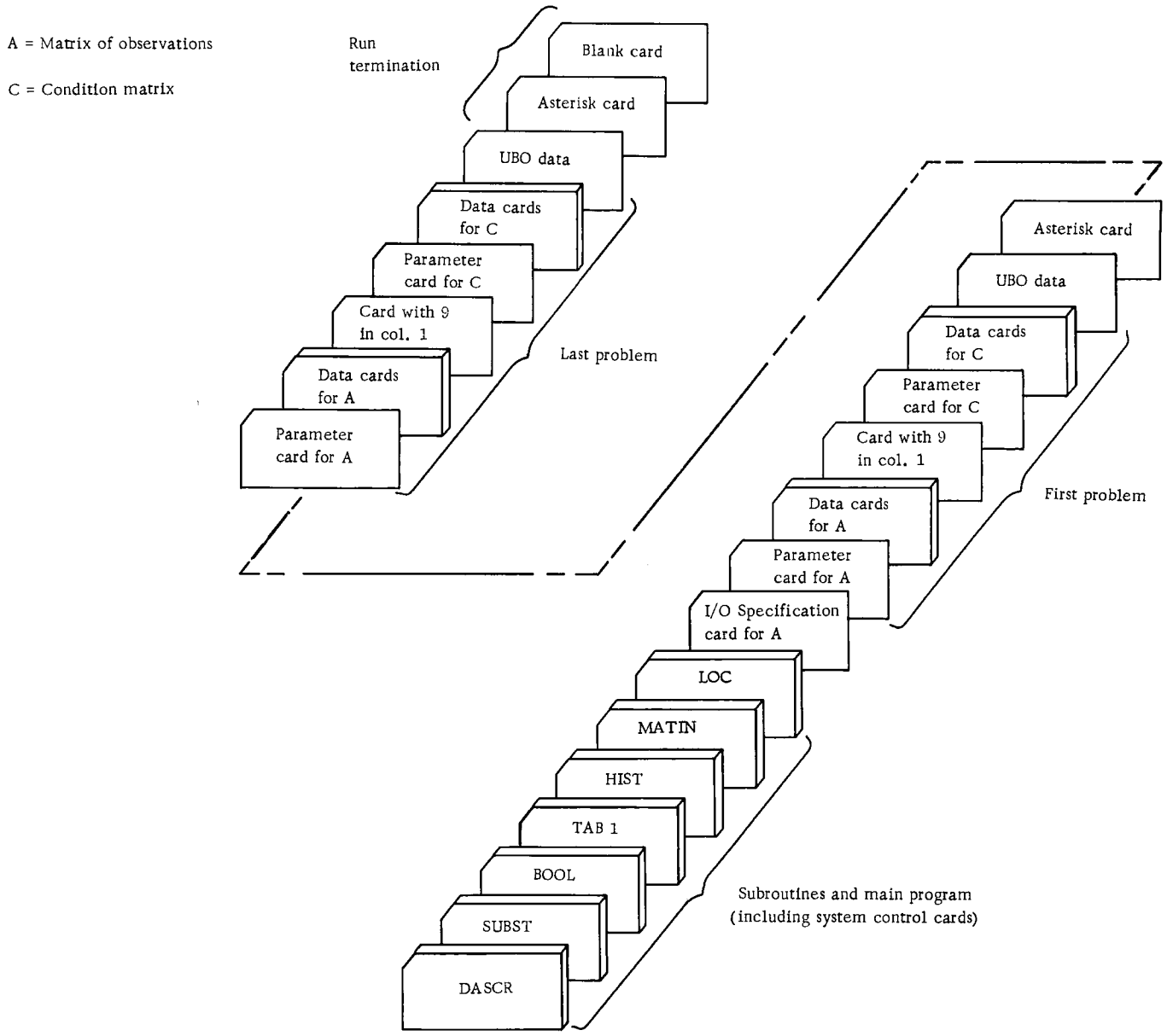
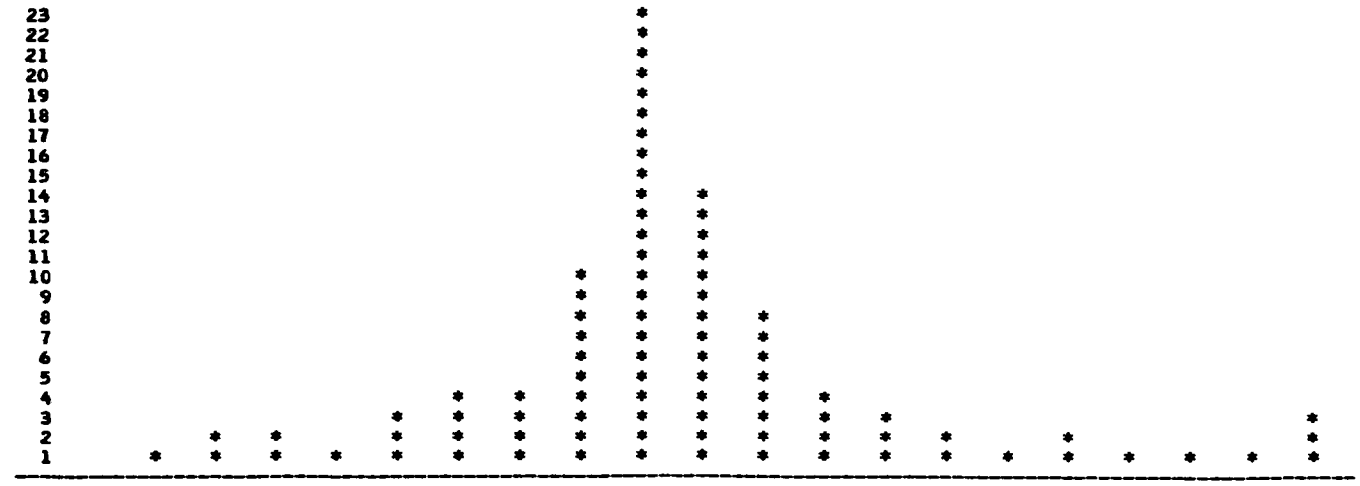


Figure 10. Deck setup (data screening)

```
DATA SCREENING PROBLEM  1

SUBSET VECTOR


  1   1.
  2   0.0
  3   1.
  4   1.
  5   1.
  6   1.
  7   1.
  8   1.
  9   1.
 10   1.
 11   1.
 12   0.0
 13   1.
 14   1.
 15   0.0
 16   1.
 17   1.
 18   1.
 19   1.
 20   1.
 21   1.
 22   1.
 23   1.
 24   1.
 25   0.0
 26   1.
 27   1.
 28   1.
 29   1.
 30   1.
 31   1.
 32   1.
 33   1.
 34   0.0
 35   1.
 36   1.
 37   1.
 38   1.
 39   1.
 40   1.
 41   0.0
 42   1.
 43   1.
 44   1.
 45   1.
 46   1.
 47   1.
 48   1.
 49   1.
 50   1.
 51   0.0
 52   1.
 53   1.
 54   1.
 55   1.
 56   1.
 57   1.
 58   1.
 59   1.
 60   1.
 61   1.
 62   1.
 63   1.
 64   1.
 65   1.
 66   1.
 67   1.
 68   1.
 69   1.
 70   1.
 71   1.
 72   1.
 73   1.
 74   1.
 75   1.
 76   1.
 77   1.
 78   0.0
 79   1.
 80   1.
 81   1.
 82   1.
 83   1.
 84   1.
 85   1.
 86   1.
 87   0.0
 88   0.0
 89   1.
 90   1.
 91   1.
 92   1.
 93   1.
 94   1.
 95   1.
 96   1.
 97   1.
 98   1.
 99   1.
100   1.

SUMMARY STATISTICS FOR VARIABLE   3

TOTAL = 14492.000  AVERAGE =   161.022  STANDARD DEVIATION =    19.329  MINIMUM =   114.000  MAXIMUM =   225.000

                                HISTOGRAM   1

FREQUENCY   1   2   2   1   3   4   4  10  23  14   8   4   3   2   1   2   1   1   1   3
```




```
INTERVAL
CLASS       1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20

END OF CASE
```

Figure 11. Output listing

Operating Instructions

The sample program for data screening is a standard FORTRAN program. Special operating instructions are not required. Logical unit 2 is used for input, and logical unit 1 is used for output.

Error Messages

The following error conditions will result in messages:

1. Reserved storage area is too small for matrix: DIMENSIONED AREA TOO SMALL FOR INPUT MATRIX. GO ON TO NEXT CASE.

2. Number of data cards does not correspond to that required by parameter card: INCORRECT NUMBER OF DATA CARDS FOR MATRIX. EXECUTION TERMINATED.

Error condition 1 allows the computer run to continue. Error condition 2, however, terminates execution and requires another run to process succeeding cases.

Sample Main Program for Data Screening - DASCR

Purpose:

Perform data screening calculations on a set of observations.

Remarks:

I/O specifications transmitted to subroutines by COMMON.

Input Card:

Column 2 MX - Logical unit number for output.

Column 4 MY - Logical unit number for input.

Subroutines and function subprograms required:

SUBST
TAB1
LOC
BOOL
HIST
MATIN

Method:

Derive a subset of observations satisfying certain conditions on the variables. For this subset, the frequency of a selected variable over given class intervals is obtained. This is plotted in the form of a histogram. Total, average, standard deviation, minimum, and maximum are also calculated.

```
// FOR
*IOCS(CARD,TYPEWRITER,1132 PRINTER)
*ONE WORD INTEGERS
C        SAMPLE MAIN PROGRAM FOR DATA SCREENING - DASCR                  DASCR  1
                                                                         DASCR  2
      EXTERNAL BOOL                                                      DASCR  3
C        THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE    DASCR  4
C        MAXIMUM NUMBER OF ELEMENTS OF THE OBSERVATION MATRIX.           DASCR  5
      DIMENSION A(1000)                                                  DASCR  6
C        THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE    DASCR  7
C        NUMBER OF CONDITIONS TIMES 3.                                   DASCR  8
      DIMENSION C(63)                                                    DASCR  9
C        THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO 3.     DASCR 10
      DIMENSION UBD(3)                                                   DASCR 11
C        THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE    DASCR 12
C        NUMBER OF OBSERVATIONS.                                         DASCR 13
      DIMENSION S(200)                                                   DASCR 14
C        THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE    DASCR 15
C        NUMBER OF CONDITIONS.                                           DASCR 16
      DIMENSION R(21)                                                    DASCR 17
C        THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO THE   DASCR 18
C        NUMBER OF INTERVALS FOR THE SELECTED VARIABLE.                  DASCR 19
      DIMENSION FREQ(20),PCT(20)                                         DASCR 20
C        THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO 5.     DASCR 21
      DIMENSION STATS(5)                                                 DASCR 22
      COMMON MX,MY                                                       DASCR 23
   10 FORMAT(////23H DATA SCREENING PROBLEM,I3)                          DASCR 24
   11 FORMAT(//45H DIMENSIONED AREA TOO SMALL FOR INPUT MATRIX ,I4)      DASCR 25
   12 FORMAT(//21H EXECUTION TERMINATED)                                 DASCR 26
   13 FORMAT(//43H INCORRECT NUMBER OF DATA CARDS FOR MATRIX ,I4)        DASCR 27
   14 FORMAT(//19H GO ON TO NEXT CASE)                                   DASCR 28
   15 FORMAT(//12H END OF CASE)                                          DASCR 29
   16 FORMAT(7(F2.0,F1.0,F7.0))                                          DASCR 30
   17 FORMAT(3F10.0)                                                     DASCR 31
   18 FORMAT(//14H SUBSET VECTOR,///)                                    DASCR 32
   19 FORMAT(16,F5.0)                                                    DASCR 33
   20 FORMAT(////33H SUMMARY STATISTICS FOR VARIABLE ,I3)                DASCR 34
   21 FORMAT(//8H TOTAL =,F10.3,2X,9HAVERAGE =,F10.3,2X,20HSTANDARD DEVI DASCR 35
     1ATION =,F10.3,2X,9HMINIMUM =,F10.3,2X,9HMAXIMUM =,F10.3)           DASCR 36
   22 FORMAT(212)                                                        DASCR 37
      KC=0                                                               DASCR 38
C        READ I/O UNIT NUMBERS                                           DASCR 39
      READ(2,22)MX,MY                                                    DASCR 40
   24 KC=KC+1                                                            DASCR 41
      CALL MATIN(ICOD,A,1000,NO,NV,MS,IER)                               DASCR 42
      IF(NO) 25,50,25                                                    DASCR 43
   25 IF(IER-1) 40,30,35                                                 DASCR 44
   30 WRITE(MX,11) ICOD                                                  DASCR 45
      WRITE(MX,14)                                                       DASCR 46
      GO TO 24                                                           DASCR 47
   35 WRITE(MX,13)                                                       DASCR 48
      WRITE(MX,12)                                                       DASCR 49
      GO TO 60                                                           DASCR 50
   40 READ(MY,22)NC,NOVAR                                                DASCR 51
      JC=NC*3                                                            DASCR 52
      READ(MY,16)(C(I),I=1,JC)                                           DASCR 53
      READ(MY,17)(UBD(I),I=1,3)                                          DASCR 54
      CALL SUBST(A,C,R,BOOL,S,NO,NV,NC)                                  DASCR 55
      WRITE(MX,10)KC                                                     DASCR 56
      WRITE(MX,18)                                                       DASCR 57
      DO 50 I=1,NO                                                       DASCR 58
   50 WRITE(MX,19)I,S(I)                                                 DASCR 59
      CALL TAB1(A,S,NOVAR,UBD,FREQ,PCT,STATS,NO,NV)                      DASCR 60
      WRITE(MX,20) NOVAR                                                 DASCR 61
      WRITE(MX,21)(STATS(I),I=1,5)                                       DASCR 62
      JZ=UBD(2)                                                          DASCR 63
      CALL HIST(KC,FREQ,JZ)                                              DASCR 64
      WRITE(MX,15)                                                       DASCR 65
      GO TO 24                                                           DASCR 66
   60 STOP                                                               DASCR 67
      END
// DUP
*STORE      WS  UA  DASCR
// XEQ DASCR
```

```
1 2                                                                      1
 000101000004                                                            2
       46        64        173       12                                 3
       24        72        170        8                                 4
       32        71        154       16                                 5
       41        68        129       10                                 6
       50        65        192        9                                 7
       63        75        203       12                                 8
       29        70        122       14                                 9
       28        64        136       13                                10
       52        77        147       11                                11
       36        67        153       18                                12
       31        68        165        9                                13
       72        70        178       10                                14
       53        71        205       14                                15
       21        65        219       12                                16
       49        63        150        6                                17
       28        62        160       16                                18
       53        72        161       13                                19
       47        73        142       15                                20
       37        67        193       18                                21
       64        68        156       14                                22
       65        60        114       10                                23
       62        64        153       12                                24
       19        68        225        9                                25
       46        67        158       11                                26
       33        72        121        4                                27
       37        65        132       13                                28
       41        76        148       16                                29
       52        71        123       16                                30
       29        68        128       14                                31
       32        65        155       17                                32
       24        72        172       16                                33
       56        73        163       10                                34
       63        65        158       11                                35
       67        69        146        2                                36
       58        66        171        9                                37
       41        65        153       12                                38
       49        66        165       14                                39
       52        72        172       16                                40
       23        78        183       15                                41
       56        71        195       16                                42
       52        68        118        7                                43
       40        66        165       14                                44
       39        68        215       16                                45
       23        71        154       12                                46
       56        65        149       10                                47
       25        65        162       16                                48
       37        68        152       16                                49
       46        70        159       15                                50
       41        69        137       14                                51
```

```
        62          71          163         12                                  52
        29          72          191          4                                  53
        19          68          168         10                                  54
        46          63          158         16                                  55
        37          64          139         18                                  56
        34          68          156         10                                  57
        64          67          153         12                                  58
        57          67          141         13                                  59
        32          68          157         17                                  60
        29          70          183         15                                  61
        53          72          164         18                                  62
        47          72          156         18                                  63
        56          73          160         16                                  64
        61          74          169         12                                  65
        21          68          161         10                                  66
        25          76          178         11                                  67
        23          72          157         16                                  68
        29          68          186         16                                  69
        39          70          159         14                                  70
        42          70          154         10                                  71
        56          62          159         12                                  72
        63          70          177         12                                  73
        51          71          161          9                                  74
        41          66          158         10                                  75
        33          69          158         16                                  76
        37          68          157         16                                  77
        25          70          163         15                                  78
        63          68          159         12                                  79
        53          71          202          6                                  80
        51          72          167         14                                  81
        47          73          164         14                                  82
        39          75          151         12                                  83
        28          68          166         10                                  84
        64          69          156         16                                  85
        55          67          144         16                                  86
        51          66          177         10                                  87
        46          65          157         12                                  88
        72          66          125         10                                  89
        66          65          131         12                                  90
        28          74          149         18                                  91
        27          71          168         11                                  92
        23          72          158         12                                  93
        23          72          163         12                                  94
        60          68          157          9                                  95
        30          66          142         10                                  96
        39          67          162         16                                  97
        46          74          154         16                                  98
        50          68          158         10                                  99
        61          66          161         14                                  100
        36          64          157         15                                  101
        32          71          156         16                                  102
9                                                                               103
0203                                                                            104
 12     65 46        8                                                         105
       120           20          210                                           106
*                                                                               107
                                                                                108
```

USER-SUPPLIED SPECIAL SUBROUTINE - BOOL

THIS SPECIAL SUBROUTINE ILLUSTRATES AN EXTERNAL SUBROUTINE CALLED BY SUBROUTINE SUBST.

IF DIFFERENT PROPOSITIONS ARE USED FOR DIFFERENT PROBLEMS IN THE SAME RUN, DIFFERENT SUBROUTINES WITH APPROPRIATE PROPOSITIONS MUST BE COMPILED UNDER DIFFERENT NAMES. IF SO, THESE SUBROUTINE NAMES MUST BE DEFINED BY AN EXTERNAL STATEMENT APPEARING IN THE MAIN PROGRAM WHICH CALLS SUBST. THEN, FOR EACH PROBLEM, SUBST IS CALLED WITH A PROPER SUBROUTINE NAME IN ITS ARGUMENT LIST.

```
      SUBROUTINE BOOL(R,T)                                              BOOL   1
      DIMENSION R(1)                                                    BOOL   2
      L1=1                                                              BOOL M01
      L2=2                                                              BOOL M02
      T=R(L1)*R(L2)                                                     BOOL M03
      RETURN                                                            BOOL   4
      END                                                               BOOL   5
```

SUBROUTINE HIST

PURPOSE
PRINT A HISTOGRAM OF FREQUENCIES VERSUS INTERVALS

USAGE
CALL HIST(NU,FREQ,IN)

DESCRIPTION OF PARAMETERS
NU - HISTOGRAM NUMBER (3 DIGITS MAXIMUM)
FREQ - VECTOR OF FREQUENCIES
IN - NUMBER OF INTERVALS AND LENGTH OF FREQ (MAX IS 20)
NORMALLY, FREQ(1) CONTAINS THE FREQUENCY SMALLER THAN THE LOWER BOUND AND FREQ(IN) CONTAINS THE FREQUENCY LARGER THAN THE UPPER BOUND

REMARKS
FREQUENCIES MUST BE POSITIVE NUMBERS

SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
NONE

METHOD
THE LARGEST FREQUENCY IS DETERMINED AND SCALING IS USED IF REQUIRED

```
      SUBROUTINE HIST(NU,FREQ,IN)                                       HIST   1
      DIMENSION JOUT(20),FREQ(20)                                       HIST   2
      COMMON MX,MY                                                      HIST   3
    1 FORMAT(6H EACH ,A1,8H EQUALS ,I2,7H POINTS,/)                     HIST   4
    2 FORMAT(1H,4X,20(1X,A1))                                           HIST   5
    3 FORMAT(//9H INTERVAL,4X,19(I2,3X),I2)                             HIST   6
    4 FORMAT(///,47X,11H HISTOGRAM ,I3)                                 HIST   7
    5 FORMAT(//10H FREQUENCY,20I5)                                      HIST   8
    6 FORMAT(6H CLASS)                                                  HIST   9
    7 FORMAT(113H   ----------                                          HIST  10
     1----------)                                                       HIST  11
    8 FORMAT(2A1)                                                       HIST  12
      READ(MY,8) K,NOTH                                                 HIST  13
C       PRINT TITLE AND FREQUENCY VECTOR                                HIST  14
      WRITE(MX,4) NU                                                    HIST  15
      DO 12 I=1,IN                                                      HIST  16
   12 JOUT(I)=FREQ(I)                                                   HIST  17
      WRITE(MX,5)(JOUT(I),I=1,IN)                                       HIST  18
      WRITE(MX,7)                                                       HIST  19
C       FIND LARGEST FREQUENCY                                          HIST  20
      FMAX=0.0                                                          HIST  21
      DO 20 I=1,IN                                                      HIST  22
      IF(FREQ(I)-FMAX) 20,20,15                                         HIST  23
   15 FMAX=FREQ(I)                                                      HIST  24
   20 CONTINUE                                                          HIST  25
C       SCALE IF NECESSARY                                              HIST  26
      JSCAL=1                                                           HIST  27
      IF(FMAX-50.0) 40,40,30                                            HIST  28
   30 JSCAL=(FMAX+49.0)/50.0                                            HIST  29
      WRITE(MX,1)K,JSCAL                                                HIST  30
C       CLEAR OUTPUT AREA TO BLANKS                                     HIST  31
   40 DO 50 I=1,IN                                                      HIST  32
   50 JOUT(I)=NOTH                                                      HIST  33
C       LOCATE FREQUENCIES IN EACH INTERVAL                             HIST  34
      MAX=FMAX/FLOAT(JSCAL)                                             HIST  35
      DO 80 I=1,MAX                                                     HIST  36
      X=MAX-(I-1)                                                       HIST  37
      DO 70 J=1,IN                                                      HIST  38
      IF(FREQ(J)/FLOAT(JSCAL)-X) 70,60,60                               HIST  39
   60 JOUT(J)=K                                                         HIST  40
   70 CONTINUE                                                          HIST  41
      IX=X*FLOAT(JSCAL)                                                 HIST  42
C       PRINT LINE OF FREQUENCIES                                       HIST  43
   80 WRITE(MX,2)IX,(JOUT(J),J=1,IN)                                    HIST  44
C       GENERATE CONSTANTS                                              HIST  45
      DO 90 I=1,IN                                                      HIST  46
   90 JOUT(I)=I                                                         HIST  47
C       PRINT INTERVAL NUMBERS                                          HIST  48
      WRITE(MX,7)                                                       HIST  49
      WRITE(MX,3)(JOUT(J),J=1,IN)                                       HIST  50
      WRITE(MX,6)                                                       HIST  51
      RETURN                                                            HIST  52
      END                                                               HIST  53
```

SUBROUTINE MATIN

PURPOSE
READS CONTROL CARD AND MATRIX DATA ELEMENTS FROM LOGICAL UNIT 5

USAGE
CALL MATIN(ICODE,A,ISIZE,IROW,ICOL,IS,IER)

DESCRIPTION OF PARAMETERS
ICODE-UPON RETURN, ICODE WILL CONTAIN FOUR DIGIT IDENTIFICATION CODE FROM MATRIX PARAMETER CARD
A -DATA AREA FOR INPUT MATRIX
ISIZE-NUMBER OF ELEMENTS DIMENSIONED BY USER FOR AREA A
IROW -UPON RETURN, IROW WILL CONTAIN ROW DIMENSION FROM MATRIX PARAMETER CARD
ICOL -UPON RETURN, ICOL WILL CONTAIN COLUMN DIMENSION FROM MATRIX PARAMETER CARD
IS -UPON RETURN, IS WILL CONTAIN STORAGE MODE CODE FROM MATRIX PARAMETER CARD WHERE
IS=0 GENERAL MATRIX
IS=1 SYMMETRIC MATRIX
IS=2 DIAGONAL MATRIX
IER -UPON RETURN, IER WILL CONTAIN AN ERROR CODE WHERE
IER=0 NO ERROR
IER=1 ISIZE IS LESS THAN NUMBER OF ELEMENTS IN INPUT MATRIX
IER=2 INCORRECT NUMBER OF DATA CARDS

REMARKS
NONE

SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
LOC

METHOD
SUBROUTINE ASSUMES THAT INPUT MATRIX CONSISTS OF PARAMETER CARD FOLLOWED BY DATA CARDS
PARAMETER CARD HAS THE FOLLOWING FORMAT
COL. 1- 2 BLANK
COL. 3- 6 UP TO FOUR DIGIT IDENTIFICATION CODE
COL. 7-10 NUMBER OF ROWS IN MATRIX
COL.11-14 NUMBER OF COLUMNS IN MATRIX
COL.15-16 STORAGE MODE OF MATRIX WHERE
0 - GENERAL MATRIX
1 - SYMMETRIC MATRIX
2 - DIAGONAL MATRIX
DATA CARDS ARE ASSUMED TO HAVE SEVEN FIELDS OF TEN COLUMNS EACH. DECIMAL POINT MAY APPEAR ANYWHERE IN A FIELD. IF NO DECIMAL POINT IS INCLUDED, IT IS ASSUMED THAT THE DECIMAL POINT IS AT THE END OF THE 10 COLUMN FIELD. NUMBER IN EACH FIELD MAY BE PRECEDED BY BLANKS. DATA ELEMENTS MUST BE PUNCHED BY ROW. A ROW MAY CONTINUE FROM CARD TO CARD. HOWEVER EACH NEW ROW MUST START IN THE FIRST FIELD OF THE NEXT CARD. ONLY THE UPPER TRIANGULAR PORTION OF A SYMMETRIC OR THE DIAGONAL ELEMENTS OF A DIAGONAL MATRIX ARE CONTAINED ON DATA CARDS. THE FIRST ELEMENT OF EACH NEW ROW WILL BE THE DIAGONAL ELEMENT FOR A MATRIX WITH SYMMETRIC OR DIAGONAL STORAGE MODE. COLUMNS 71-80 OF DATA CARDS MAY BE USED FOR IDENTIFICATION, SEQUENCE NUMBERING, ETC.. THE LAST DATA CARD FOR ANY MATRIX MUST BE FOLLOWED BY A CARD WITH A 9 PUNCH IN COLUMN 1.

```
      SUBROUTINE MATIN(ICODE,    A,ISIZE,IROW,ICOL,IS,IER)              MATIN   1
      DIMENSION A(1)                                                    MATIN   2
      DIMENSION CARD(8)                                                 MATIN   3
      COMMON MX,MY                                                      MATIN   4
    1 FORMAT(7F10.0)                                                    MATIN   5
    2 FORMAT(I6,2I4,I2)                                                 MATIN   6
    3 FORMAT(I1)                                                        MATIN   7
      IDC=7                                                             MATIN   8
      IER=0                                                             MATIN   9
      READ( MY,2)ICODE,IROW,ICOL,IS                                     MATIN  10
      CALL LOC(IROW,ICOL,ICNT,IROW,ICOL,IS)                             MATIN  11
      IF(ISIZE-ICNT)6,7,7                                               MATIN  12
    6 IER=1                                                             MATIN  13
    7 IF (ICNT)38,38,8                                                  MATIN  14
    8 ICOLT=ICOL                                                        MATIN  15
      IROCR=1                                                           MATIN  16
C       COMPUTE NUMBER OF CARDS FOR THIS ROW                            MATIN  17
   11 IRCDS=(ICOLT-1)/IDC+1                                             MATIN  18
      IF(IS-1)15,15,12                                                  MATIN  19
   12 IRCDS=1                                                           MATIN  20
C       SET UP LOOP FOR NUMBER OF CARDS IN ROW                          MATIN  21
   15 DO 31 K=1,IRCDS                                                   MATIN  22
      READ(MY,1)(CARD(I),I=1,IDC)                                       MATIN  23
C       SKIP THROUGH DATA CARDS IF INPUT AREA TOO SMALL                 MATIN  24
      IF(IER)16,16,31                                                   MATIN  25
   16 L=0                                                               MATIN  26
C       COMPUTE COLUMN NUMBER FOR FIRST FIELD IN CURRENT CARD           MATIN  27
      JS=(K-1)*IDC+ICOL-ICOLT+1                                         MATIN  28
```

```
      JE=JS+IDC-1                                                    MATIN 29
      IF(IS-1)19,19,17                                               MATIN 30
   17 JF=JS                                                          MATIN 31
C        SET UP LOOP FOR DATA ELEMENTS  WITHIN CARD                  MATIN 32
   19 DO 30 J=JS,JE                                                  MATIN 33
      IF(J-ICOL)20,20,31                                             MATIN 34
   20 CALL LOC(IROCR ,J,IJ,IROW,ICOL,IS)                             MATIN 35
      L=L+1                                                          MATIN 36
   30 A(IJ)=CARD(L)                                                  MATIN 37
   31 CONTINUE                                                       MATIN 38
      IROCR=IROCR+1                                                  MATIN 39
      IF(IROW-IROCR) 38,35,35                                        MATIN 40
   35 IF(IS-1)37,36,36                                               MATIN 41
   36 ICOLT=ICOLT-1                                                  MATIN 42
   37 GO TO 11                                                       MATIN 43
   38 READ(MY,3)ICARD                                                MATIN 44
      IF(ICARD-9)39,40,39                                            MATIN 45
   39 IER=2                                                          MATIN 46
   40 RETURN                                                         MATIN 47
      END                                                            MATIN 48
```

MULTIPLE LINEAR REGRESSION

Problem Description

Multiple linear regression analysis is performed for a set of independent variables and a dependent variable. Selection of different sets of independent variables and designation of a dependent variable can be made as many times as desired.

The sample problem for multiple linear regression consists of 30 observations with six variables as presented in Table 2. The first five variables are independent variables, and the last variable is the dependent variable. All five independent variables are used to predict the dependent variable in the first analysis, and only second, third, and fifth variables are used to predict the dependent variable in the second analysis.

Table 2. Sample Data for Multiple Linear Regression

	Variables					
Observation	X_1	X_2	X_3	X_4	X_5	X_6
1	29	289	216	85	14	1
2	30	391	244	92	16	2
3	30	424	246	90	18	2
4	30	313	239	91	10	0
5	35	243	275	95	30	2
6	35	365	219	95	21	2
7	43	396	267	100	39	3
8	43	356	274	79	19	2
9	44	346	255	126	56	3
10	44	156	258	95	28	0
11	44	278	249	110	42	4
12	44	349	252	88	21	1
13	44	141	236	129	56	1
14	44	245	236	97	24	1
15	45	297	256	111	45	3
16	45	310	262	94	20	2
17	45	151	339	96	35	3
18	45	370	357	88	15	4
19	45	379	198	147	64	4
20	45	463	206	105	31	3
21	45	316	245	132	60	4
22	45	280	225	108	36	4
23	44	395	215	101	27	1
24	49	139	220	136	59	0
25	49	245	205	113	37	4
26	49	373	215	88	25	1
27	51	224	215	118	54	3
28	51	677	210	116	33	4
29	51	424	210	140	59	4
30	51	150	210	105	30	0

Program

Description

The multiple linear regression sample program consists of a main routine, REGRE, and five subroutines:

CORRE, ORDER, MINV, MULTR	are from the Scientific Subroutine Package
DATA	is a special input subroutine

Capacity

The capacity of the sample program and the format required for data input have been set up as follows:

1. Up to 21 variables, including both independent and dependent variables.
2. Up to 99,999 observations, if observations are read into the computer one at a time by the special input subroutine named DATA. If all data are to be stored in core prior to the calculation of correlation coefficients, the limitation on the number of observations depends on the size of core storage available for input data.
3. (12F6.0) format for input data cards.

Therefore, if a problem satisfies the above conditions it is not necessary to modify the sample program. However, if there are more than 22 variables, dimension statements in the sample main program must be modified to handle this particular problem. Similarly, if input data cards are prepared using a different format, the input format in the input subroutine, DATA, must be modified. The general rules for program modification are described later.

Input

One I/O Specification card defines input/output units (see "Sample Program Descriptions").

One control card is required for each problem and is read by the main program, REGRE. This card is prepared as follows:

Columns	Contents	For Sample Problem
1 - 6	Problem number (may be alphameric)	SAMPLE

Columns	Contents	For Sample Problem
7 - 11	Number of observations	00030
12 - 13	Number of variables	06
14 - 15	Number of selection cards (see below)	02

Leading zeros are not required to be keypunched, but all numbers must be right-justified, even if a decimal point is included.

Data Cards

Since input data are read into the computer one observation at a time, each row of data in Table 2 is keypunched on a separate card using the format (12F6.0). This format assumes twelve 6-column fields per card.

If there are more than twelve variables in a problem, each row of data is continued on the second and third cards until the last data point is keypunched. However, each row of data must begin on a new card.

Selection Card

The selection card is used to specify a dependent variable and a set of independent variables in a multiple linear regression analysis. Any variable in the set of original variables can be designated as a dependent variable, and any number of variables can be specified as independent variables. Selection of a dependent variable and a set of independent variables can be performed over and over again using the same set of original variables.

The selection card is prepared as follows:

Columns	Contents	For Sample Problem Selection 1	Selection 2
1 - 2	Option code for table of residuals 00 if it is not desired 01 if it is desired	01	01
3 - 4	Dependent variable designated for the forthcoming regression	06	06
5 - 6	Number of independent variables included in the forthcoming regression	05	03
5 - 6 (cont)	(the subscript numbers of individual variables are specified below)		
7 - 8	1st independent variable included	01	02
9 - 10	2nd independent variable included	02	03
11 - 12	3rd independent variable included	03	05
13 - 14	4th independent variable included	04	
15 - 16	5th independent variable included	05	

etc.

The input format of (36I2) is used for the selection card.

Deck Setup

Deck setup is shown in Figure 12.

The repetition of the data cards following a selection card is dependent upon the option code for the table of residuals. If the table is required (option 01), the data must be repeated; if the table is not required (option 00), card G immediately follows card E.

Sample

The listing of input cards for the sample problem is presented at the end of the sample main program.

Output

Description

The output of the sample program for multiple linear regression includes:

1. Means
2. Standard deviations
3. Correlation coefficients between the independent variables and the dependent variable
4. Regression coefficients
5. Standard errors of regression coefficients
6. Computed t-values
7. Intercept
8. Multiple correlation coefficients

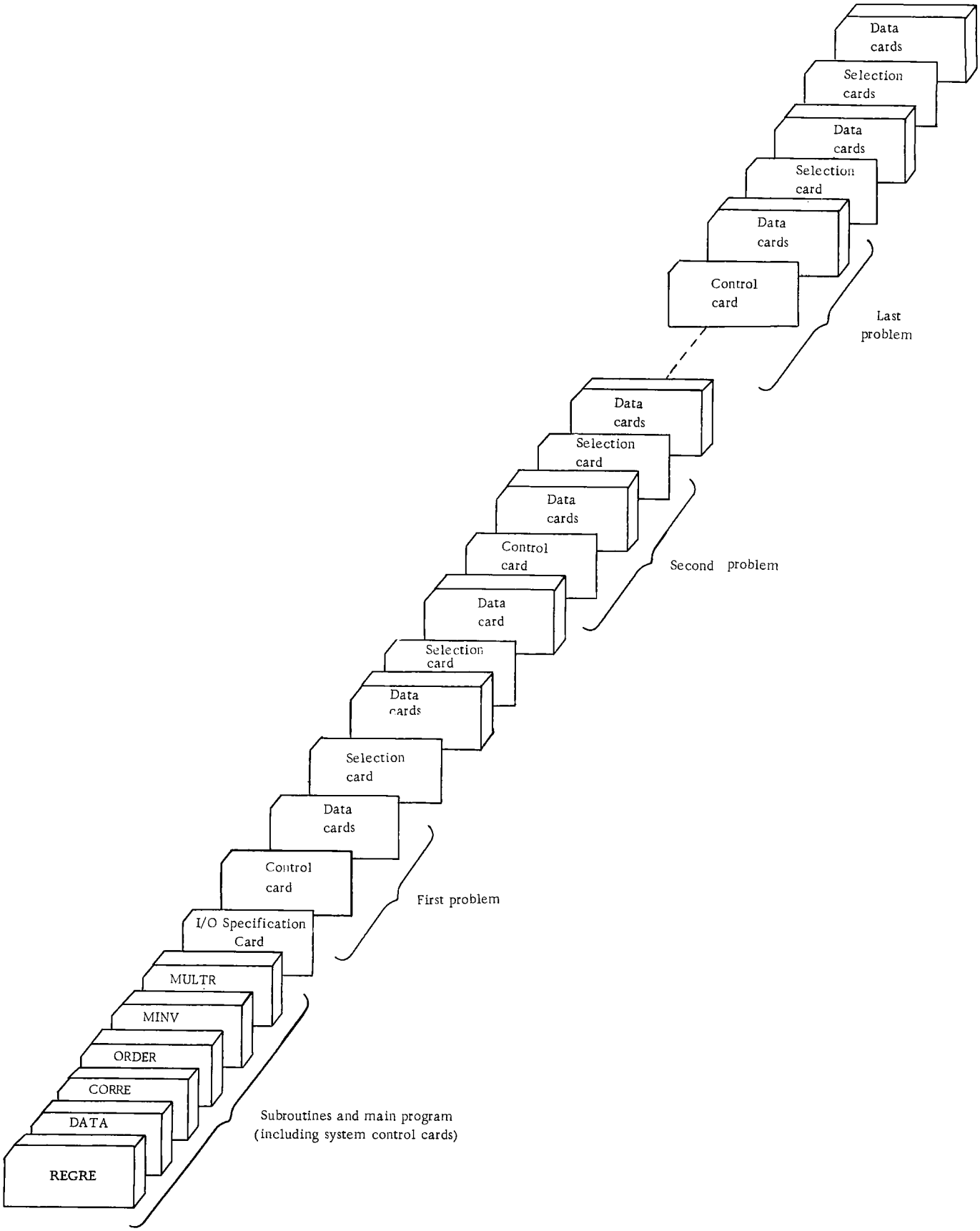


Figure 12. Deck setup (multiple linear regression)

9. Standard error of estimate
10. Analysis of variance for the multiple regression
11. Table of residuals (optional)

Sample

The output listing for the sample problem is shown in Figure 13.

Program Modification

Noting that storage problems may result, as previously discussed in "Sample Program Description", program capacity can be increased or decreased by making changes in dimension statements. Input data in a different format can also be handled by providing a specific format statement. In order to familiarize the user with the program modification, the following general rules are supplied in terms of the sample problem:

1. Changes in the dimension statements of the main program, REGRE:
   a. The dimension of arrays XBAR, STD, D, RY, ISAVE, B, SB, T, and W must be greater than or equal to the number of variables, m. Since there are six variables in the sample problem the value of m is 6.
   b. The dimension of array RX must be greater than or equal to the product of m x m. For the sample problem this product is 36 = 6 x 6.
   c. The dimension of array R must be greater than or equal to (m + 1) m/2. For the sample problem this number is 21 = (6 + 1) 6/2.

MULTIPLE REGRESSION.....SAMPLE

SELECTION..... 1

VARIABLE NO.	MEAN	STANDARD DEVIATION	CORRELATION X VS Y	REGRESSION COEFFICIENT	STD. ERROR OF REG.COEF.	COMPUTED T VALUE
1	43.13333	6.52175	0.28422	0.01242	0.03634	0.34172
2	316.16668	114.42994	0.42189	0.00738	0.00186	3.96545
3	241.80001	36.43074	0.11900	0.01504	0.00634	2.36882
4	105.66667	17.85638	0.37822	0.00150	0.03678	0.04101
5	34.13333	15.97569	0.39412	0.04918	0.04141	1.18782

DEPENDENT

6	2.26666	1.41258

INTERCEPT -6.07935

MULTIPLE CORRELATION 0.73575

STD. ERROR OF ESTIMATE 1.05161

ANALYSIS OF VARIANCE FOR THE REGRESSION

SOURCE OF VARIATION	DEGREES OF FREEDOM	SUM OF SQUARES	MEAN SQUARES	F VALUE
ATTRIBUTABLE TO REGRESSION	5	31.32517	6.26503	5.66512
DEVIATION FROM REGRESSION	24	26.54149	1.10589	
TOTAL	29	57.86666		

MULTIPLE REGRESSION.....SAMPLE

SELECTION..... 1

TABLE OF RESIDUALS

CASE NO.	Y VALUE	Y ESTIMATE	RESIDUAL
1	1.00000	0.48089	0.51910
2	2.00000	1.77669	0.22330
3	2.00000	2.14585	-0.14585
4	0.00000	0.82879	-0.82879
5	2.00000	1.90522	0.09477
6	2.00000	1.52124	0.47875
7	3.00000	3.46447	-0.46447
8	2.00000	2.25886	-0.25886
9	3.00000	3.80259	-0.80259
10	0.00000	1.02042	-1.02042
11	4.00000	2.49735	1.50264
12	1.00000	2.00065	-1.00065
13	1.00000	2.00735	-1.00735
14	1.00000	1.15307	-0.15307
15	3.00000	2.90445	0.09554
16	2.00000	1.83531	0.16468
17	3.00000	2.56004	0.43995
18	4.00000	3.45229	0.54770
19	4.00000	3.62661	0.37338
20	3.00000	2.68067	0.31932
21	4.00000	3.64886	0.35113
22	4.00000	1.86541	2.13458
23	1.00000	2.09862	-1.09862
24	0.00000	1.97217	-1.97217
25	4.00000	1.41253	2.58746
26	1.00000	1.88026	-0.88026
27	3.00000	2.27546	0.72353
28	4.00000	4.51080	-0.51080
29	4.00000	3.95746	0.04253
30	0.00000	0.45457	-0.45457

MULTIPLE REGRESSION.....SAMPLE

SELECTION..... 2

VARIABLE NO.	MEAN	STANDARD DEVIATION	CORRELATION X VS Y	REGRESSION COEFFICIENT	STD. ERROR OF REG.COEF.	COMPUTED T VALUE
2	316.16668	114.42994	0.42189	0.00743	0.00172	4.31764
3	241.80001	36.43074	0.11900	0.01497	0.00591	2.71693
5	34.13333	15.97569	0.39412	0.05362	0.01258	4.26263

DEPENDENT

6	2.26666	1.41258

INTERCEPT -5.53530

MULTIPLE CORRELATION 0.73423

STD. ERROR OF ESTIMATE 1.01281

ANALYSIS OF VARIANCE FOR THE REGRESSION

SOURCE OF VARIATION	DEGREES OF FREEDOM	SUM OF SQUARES	MEAN SQUARES	F VALUE
ATTRIBUTABLE TO REGRESSION	3	31.19601	10.39867	10.13718
DEVIATION FROM REGRESSION	26	26.67065	1.02579	
TOTAL	29	57.86666		

MULTIPLE REGRESSION.....SAMPLE

SELECTION..... 2

TABLE OF RESIDUALS

CASE NO.	Y VALUE	Y ESTIMATE	RESIDUAL
1	1.00000	0.59868	0.40131
2	2.00000	1.88362	0.11637
3	2.00000	2.26619	-0.26619
4	0.00000	0.90703	-0.90703
5	2.00000	1.99812	0.00187
6	2.00000	1.58407	0.41592
7	3.00000	3.49859	-0.49858
8	2.00000	2.23347	-0.23347
9	3.00000	3.85875	-0.85875
10	0.00000	0.98943	-0.98943
11	4.00000	2.51254	1.48745
12	1.00000	1.95925	-0.95925
13	1.00000	2.04998	-1.04998
14	1.00000	1.10725	-0.10725
15	3.00000	2.91951	0.08048
16	2.00000	1.76538	0.23461
17	3.00000	2.54052	0.45947
18	4.00000	3.36591	0.63408
19	4.00000	3.67961	0.32038
20	3.00000	2.65434	0.34565
21	4.00000	3.70045	0.29954
22	4.00000	1.84628	2.15371
23	1.00000	2.06899	-1.06899
24	0.00000	1.95640	-1.95640
25	4.00000	1.34019	2.65980
26	1.00000	1.79816	-0.79816
27	3.00000	2.24542	0.75458
28	4.00000	4.41268	-0.41268
29	4.00000	3.92577	0.07423
30	0.00000	0.33331	-0.33331

Figure 13. Output listing

2. Changes in the input format statement of the special input subroutine, DATA:

Only the format statement for input data may be changed. Since sample data are either one-, two-, or three-digit numbers, rather than using six-column fields as in the sample problem, each row of data may be keypunched in six 3-column fields, and, if so, the format is changed to (6F3.0).

The special input subroutine, DATA, is normally written by the user to handle different formats for different problems. The user may modify this subroutine to perform testing of input data, transformation of data, and so on.

Operating Instructions

The sample program for multiple linear regression is a standard FORTRAN program. Special operating instructions are not required. Logical unit 2 is used for input, and logical unit 1 is used for output.

Error Messages

The following error conditions will result in messages:

1. The number of selection cards is not specified on the control card: NUMBER OF SELECTIONS NOT SPECIFIED. JOB TERMINATED.

2. The matrix of correlation coefficients is singular: THE MATRIX IS SINGULAR. THIS SELECTION IS SKIPPED.

Error condition 2 allows the computer run to continue; however, error condition 1 terminates execution of the job.

Sample Main Program for Multiple Regression - REGRE

Purpose:

(1) Read the problem parameter card for a multiple regression, (2) Read subset selection cards, (3) Call the subroutines to calculate means, standard deviations, simple and multiple correlation coefficients, regression coefficients. T-values, and analysis of variance for multiple regression, and (4) Print the results.

Remarks:

The number of observations, N, must be greater than M+1, where M is the number of variables. If subset selection cards are not present, the program can not perform multiple after returning from subroutine MINV, the value of determinant (DET) is tested to check whether the correlation matrix is singular. If DET is compared against a small constant, this test may also be used to check near-singularity.

I/O specifications transmitted to subroutines by COMMON.

Input card:

Column 2 MX - Logical unit number for output.

Column 4 MY - Logical unit number for input.

Subroutines and function subprograms required:

CORRE (which, in turn, calls the subroutine named DATA)

ORDER

MINV

MULTR

Method:

Refer to B. Ostle, 'Statistics in Research', The Iowa State College Press', 1954, Chapter 8.

```
// FOR
*IOCS(CARD,TYPEWRITER,1132 PRINTER)
*ONE WORD INTEGERS
C        SAMPLE MAIN PROGRAM FOR MULTIPLE REGRESSION - REGRE                    REGRE  1
C        THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO THE          REGRE  2
C        NUMBER OF VARIABLES,M.                                                 REGRE  3
      DIMENSION XBAR(21),STD(21),D(21),RY(21),ISAVE(21),B(21),                  REGREMO1
     1SB(21),T(21),W(21)                                                        REGREMO2
C        THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE           REGRE  6
C        PRODUCT OF M*M.                                                        REGRE  7
      DIMENSION RX(441)                                                         REGREMO3
C        THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO               REGRE  9
C        (M+1)*M/2.                                                             REGRE 10
      DIMENSION R(231)                                                          REGREMO4
C        THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO 10.           REGRE 12
      DIMENSION ANS(10)                                                         REGRE 13
      COMMON MX,MY                                                              REGRE 14
      READ(2,15)MX,MY                                                           REGRE 15
   15 FORMAT(2I2)                                                               REGRE 16
    1 FORMAT(A6,A2,I5,2I2)                                                      REGRE 17
    2 FORMAT(////25H MULTIPLE REGRESSION.....A6,A2//6X,14HSELECTION.....REGRE 18
     1I2//)                                                                     REGRE 19
    3 FORMAT(//9H VARIABLE,5X,4HMEAN,6X,8HSTANDARD,6X,11HCORRELATION,4X,REGRE 20
     110HREGRESSION,4X,10HSTD. ERROR,5X,8HCOMPUTED/6H   NO.,18X,9HDEVIATREGRE 21
     2ION,7X,6HX VS Y,7X,11HCOEFFICIENT,3X,12HOF REG.COEF.,3X,7HT VALUE)REGRE 22
    4 FORMAT(,I4,6F14.5)                                                        REGRE 23
    5 FORMAT(//,10H DEPENDENT)                                                  REGRE 24
    6 FORMAT(///10H INTERCEPT,13X,F13.5//23H MULTIPLE CORRELATION  =F13.REGRE 25
     15//23H STD. ERROR OF ESTIMATE,F13.5//)                                    REGRE 26
    7 FORMAT(//,21X,39HANALYSIS OF VARIANCE FOR THE REGRESSION//5X,19HSOREGRE 27
     1URCE OF VARIATION,7X,7HDEGREES,7X,6HSUM OF,10X,4HMEAN,13X,7HF VALUREGRE 28
     2E/30X,10HOF FREEDOM,4X,7HSQUARES,9X,7HSQUARES)                            REGRE 29
    8 FORMAT(/30H ATTRIBUTABLE TO REGRESSION   =I6,3F16.5/30H DEVIATION REGRE 30
     1FROM REGRESSION    =I6,2F16.5)                                            REGRE 31
    9 FORMAT(/5X,5HTOTAL,19X,I6,F16.5)                                          REGRE 32
   10 FORMAT(3612)                                                              REGRE 33
   11 FORMAT(/,15X,18HTABLE OF RESIDUALS//9H CASE NO.,5X,7HY VALUE,5X,10REGRE 34
     1HY ESTIMATE,6X,8HRESIDUAL)                                                REGRE 35
   12 FORMAT(I6,F15.5,2F14.5)                                                   REGRE 36
   13 FORMAT(////53H NUMBER OF SELECTIONS NOT SPECIFIED.  JOB TERMINATEDREGRE 37
     1.)                                                                        REGRE 38
   14 FORMAT(//52H THE MATRIX IS SINGULAR.  THIS SELECTION IS SKIPPED.)         REGRE 39
C        READ PROBLEM PARAMETER CARD                                            REGRE 40
  100 READ (MY,1) PR,PR1,N,M,NS                                                 REGRE 41
C        PR........PROBLEM NUMBER (MAY BE ALPHAMERIC)                           REGRE 42
C        PR1.......PROBLEM NUMBER (CONTINUED)                                  REGRE 43
C        N.........NUMBER OF OBSERVATIONS                                       REGRE 44
C        M.........NUMBER OF VARIABLES                                          REGRE 45
C        NS........NUMBER OF SELECTIONS                                         REGRE 46
      IO=0                                                                      REGRE 47
      X=0.0                                                                     REGRE 48
      CALL CORRE (N,M,IO,X,XBAR,STD,RX,R,D,B,T)                                 REGRE 49
C        TEST NUMBER OF SELECTIONS                                              REGRE 50
      IF(NS) 108, 108, 109                                                      REGRE 51
  108 WRITE (MX,13)                                                             REGRE 52
      GO TO 300                                                                 REGRE 53
  109 DO 200 I=1,NS                                                             REGRE 54
      WRITE (MX,2) PR,PR1,I                                                     REGRE 55
C        READ SUBSET SELECTION CARD                                             REGRE 56
      READ (MY,10)NRESI,NDEP,K,(ISAVE(J),J=1,K)                                 REGRE 57
C        NRESI.....OPTION CODE FOR TABLE OF RESIDUALS                           REGRE 58
C                  0  IF IT IS NOT DESIRED.                                     REGRE 59
C                  1  IF IT IS DESIRED.                                         REGRE 60
C        NDEP......DEPENDENT VARIABLE                                           REGRE 61
C        K.........NUMBER OF INDEPENDENT VARIABLES INCLUDED                     REGRE 62
C        ISAVE.....A VECTOR CONTAINING THE INDEPENDENT VARIABLES                REGRE 63
C                  INCLUDED                                                     REGRE 64
      CALL ORDER (M,R,NDEP,K,ISAVE,RX,RY)                                       REGRE 65
      CALL MINV (RX,K,DET,B,T)                                                  REGRE 66
C        TEST SINGULARITY OF THE MATRIX INVERTED                                REGRE 67
      IF(DET) 112, 110, 112                                                     REGRE 68
  110 WRITE (MX,14)                                                             REGRE 69
      GO TO 200                                                                 REGRE 70
  112 CALL MULTR (N,K,XBAR,STD,D,RX,RY,ISAVE,B,SB,T,ANS)                        REGRE 71
C        PRINT MEANS, STANDARD DEVIATIONS, INTERCORRELATIONS BETWEEN            REGRE 72
C        X AND Y, REGRESSION COEFFICIENTS, STANDARD DEVIATIONS OF               REGRE 73
C        REGRESSION COEFFICIENTS, AND COMPUTED T-VALUES                         REGRE 74
      MM=K+1                                                                    REGRE 75
      WRITE (MX,3)                                                              REGRE 76
      DO 115 J=1,K                                                              REGRE 77
      L=ISAVE(J)                                                                REGRE 78
  115 WRITE (MX,4) L,XBAR(L),STD(L),RY(J),B(J),SB(J),T(J)                       REGRE 79
      WRITE (MX,5)                                                              REGRE 80
      L=ISAVE(MM)                                                               REGRE 81
      WRITE (MX,4) L,XBAR(L),STD(L)                                             REGRE 82
C        PRINT INTERCEPT, MULTIPLE CORRELATION COEFFICIENT, AND                 REGRE 83
C        STANDARD ERROR OF ESTIMATE                                             REGRE 84
      WRITE (MX,6) ANS(1),ANS(2),ANS(3)                                         REGRE 85
```

```
C       PRINT ANALYSIS OF VARIANCE FOR THE REGRESSION                   REGRE 86
      WRITE (MX,7)                                                      REGRE 87
      L=ANS(8)                                                          REGRE 88
      WRITE (MX,8) K,ANS(4),ANS(6),ANS(10),L,ANS(7),ANS(9)              REGRE 89
      L=N-1                                                             REGRE 90
      SUM=ANS(4)+ANS(7)                                                 REGRE 91
      WRITE (MX,9) L,SUM                                                REGRE 92
      IF(NRESI) 200, 200, 120                                           REGRE 93
C       PRINT TABLE OF RESIDUALS                                        REGRE 94
  120 WRITE(MX,2)PR,PR1,I                                               REGRE 95
      WRITE (MX,11)                                                     REGRE 96
      MM=ISAVE(K+1)                                                     REGRE 97
      DO  140 II=1,N                                                    REGRE 98
      CALL DATA(M,A)                                                    REGRE 99
      SUM=ANS(1)                                                        REGRE100
      DO 130 J=1,K                                                      REGRE101
      L=ISAVE(J)                                                        REGRE102
  130 SUM=SUM+W(L)*B(J)                                                 REGRE103
      RESI=W(MM)-SUM                                                    REGRE104
  140 WRITE (MX,12) II,W(MM),SUM,RESI                                   REGRE105
  200 CONTINUE                                                          REGRE106
      GO TO 100                                                         REGRE107
  300 STOP                                                              REGRE108
      END                                                               REGRE109
```

```
 1 2                                                                    1
SAMPLE000300602                                                         2
    29   289   216    85    14    1                                     3
    30   391   244    92    16    2                                     4
    30   424   246    90    18    2                                     5
    30   313   239    91    10    0                                     6
    35   243   275    95    30    2                                     7
    35   365   219    95    21    2                                     8
    43   396   267   100    39    3                                     9
    43   356   274    79    19    2                                    10
    44   346   255   126    56    3                                    11
    44   156   258    95    28    0                                    12
    44   278   249   110    42    4                                    13
    44   349   252    88    21    1                                    14
    44   141   236   129    56    1                                    15
    44   245   236    97    24    1                                    16
    45   297   256   111    45    3                                    17
    45   310   262    94    20    2                                    18
    45   151   339    96    35    3                                    19
    45   370   357    88    15    4                                    20
    45   379   198   147    64    4                                    21
    45   463   206   105    31    3                                    22
    45   316   245   132    60    4                                    23
    45   280   225   108    36    4                                    24
    44   395   215   101    27    1                                    25
    49   139   220   136    59    0                                    26
    49   245   205   113    37    4                                    27
    49   373   215    88    25    1                                    28
    51   224   215   118    54    3                                    29
    51   677   210   116    33    4                                    30
    51   424   210   140    59    4                                    31
    51   150   210   105    30    0                                    32
010605010203040 5                                                     33
    29   289   216    85    14    1                                    34
    30   391   244    92    16    2                                    35
    30   424   246    90    18    2                                    36
    30   313   239    91    10    0                                    37
    35   243   275    95    30    2                                    38
    35   365   219    95    21    2                                    39
    43   396   267   100    39    3                                    40
    43   356   274    79    19    2                                    41
    44   346   255   126    56    3                                    42
    44   156   258    95    28    0                                    43
    44   278   249   110    42    4                                    44
    44   349   252    88    21    1                                    45
    44   141   236   129    56    1                                    46
    44   245   236    97    24    1                                    47
    45   297   256   111    45    3                                    48
    45   310   262    94    20    2                                    49
    45   151   339    96    35    3                                    50
    45   370   357    88    15    4                                    51
    45   379   198   147    64    4                                    52
    45   463   206   105    31    3                                    53
    45   316   245   132    60    4                                    54
    45   280   225   108    36    4                                    55
    44   395   215   101    27    1                                    56
    49   139   220   136    59    0                                    57
    49   245   205   113    37    4                                    58
    49   373   215    88    25    1                                    59
    51   224   215   118    54    3                                    60
    51   677   210   116    33    4                                    61
    51   424   210   140    59    4                                    62
    51   150   210   105    30    0                                    63
010603020305                                                          64
    29   289   216    85    14    1                                    65
    30   391   244    92    16    2                                    66
    30   424   246    90    18    2                                    67
    30   313   239    91    10    0                                    68
    35   243   275    95    30    2                                    69
    35   365   219    95    21    2                                    70
    43   396   267   100    39    3                                    71
    43   356   274    79    19    2                                    72
    44   346   255   126    56    3                                    73
    44   156   258    95    28    0                                    74
    44   278   249   110    42    4                                    75
    44   349   252    88    21    1                                    76
    44   141   236   129    56    1                                    77
    44   245   236    97    24    1                                    78
    45   297   256   111    45    3                                    79
    45   310   262    94    20    2                                    80
    45   151   339    96    35    3                                    81
    45   370   357    88    15    4                                    82
    45   379   198   147    64    4                                    83
    45   463   206   105    31    3                                    84
    45   316   245   132    60    4                                    85
    45   280   225   108    36    4                                    86
    44   395   215   101    27    1                                    87
    49   139   220   136    59    0                                    88
    49   245   205   113    37    4                                    89
    49   373   215    88    25    1                                    90
    51   224   215   118    54    3                                    91
    51   677   210   116    33    4                                    92
    51   424   210   140    59    4                                    93
    51   150   210   105    30    0                                    94
```

```
      SAMPLE INPUT SUBROUTINE - DATA

      PURPOSE
         READ AN OBSERVATION (M DATA VALUES) FROM INPUT DEVICE.
         THIS SUBROUTINE IS CALLED BY THE SUBROUTINE CORRE AND MUST
         BE PROVIDED BY THE USER.  IF SIZE AND LOCATION OF DATA
         FIELDS ARE DIFFERENT FROM PROBLEM TO PROBLEM, THIS SUB-
         ROUTINE MUST BE RECOMPILED WITH A PROPER FORMAT STATEMENT.

      USAGE
         CALL DATA (M,D)

      DESCRIPTION OF PARAMETERS
         M - THE NUMBER OF VARIABLES IN AN OBSERVATION.
         D - OUTPUT VECTOR OF LENGTH M CONTAINING THE OBSERVATION
             DATA.

      REMARKS
         THE TYPE OF CONVERSION SPECIFIED IN THE FORMAT MUST BE
         EITHER F OR E.

      SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
         NONE
```

```
      SUBROUTINE DATA (M,D)                                             DATA  1
      DIMENSION D(1)                                                    DATA  2
      COMMON MX,MY                                                      DATA  3
    1 FORMAT(12F6.0)                                                    DATA  4
C       READ AN OBSERVATION FROM INPUT DEVICE.                          DATA  5
      READ (MY,1) (D(I),I=1,M)                                          DATA  6
      RETURN                                                            DATA  7
      END                                                               DATA  8
```

POLYNOMIAL REGRESSION

Problem Description

Powers of an independent variable are generated to calculate polynomials of successively increasing degrees. If there is no reduction in the residual sum of squares between two successive degrees of polynomials, the program terminates the problem before completing the analysis for the highest degree polynomial specified.

The sample problem for polynomial regression consists of 15 observations, as presented in Table 3. The highest degree polynomial specified for this problem is 4.

Table 3. Sample Data for Polynomial Regression

X	Y
1	10
2	16
3	20
4	23
5	25
6	26
7	30
8	36
9	48
10	62
11	78
12	94
13	107
14	118
15	127

Program

Description

The polynomial regression sample program consists of a main routine, POLRG, and five subroutines:

GDATA ORDER MINV MULTR	are from the Scientific Subroutine Package
PLOT	is a special plot subroutine

Capacity

The capacity of the sample program and the format required for data input have been set up as follows:

1. Up to 50 observations
2. Up to 6th degree polynomials
3. (2F 6.0) format for input data cards

Therefore, if a problem satisfies the above conditions it is not necessary to modify the sample program. However, if there are more than 60 observations or if greater than 7th degree polynomial is desired, dimension statements in the sample main program must be modified to handle this particular problem. Similarly, if input data cards are prepared using a different format, the input format in the sample main program must be modified. The general rules for program modification are described later.

Input

I/O Specification Card

One control card is required for each problem and is read by the main program, POLRG. This card is prepared as follows:

Columns	Contents	For Sample Problem
1 - 6	Problem number (may be alphameric)	SAMPLE
7 - 11	Number of observations	00015
12 - 13	Highest degree polynomial to be fitted	04
14	Option code for plotting Y values and Y estimates: 0 if it is not desired 1 if it is desired	1

Leading zeros are not required to be keypunched; but numbers must be right-justified in fields.

Data Cards

Since input data are read into the computer one observation at a time, each pair of X and Y data in Table 3 is keypunched in that order on a separate card using the format (2F 6.0).

Plot Option Card

A card containing b12...9 (blank followed by numbers 1 through 9) in columns 1 to 10 is necessary after each set of data if plotting is required (option 1). If plotting is not required (option 0), this card must be omitted.

Deck Setup

Deck setup is shown in Figure 14.

Sample

The listing of input cards for the sample problem is presented at the end of the sample main program.

Output

Description

The output of the sample program for polynomial regression includes:

1. Regression coefficients for successive degree polynomial
2. Analysis-of-variance table for successive degree polynomial
3. Table of residuals for the final degree polynomial (included with plot)
4. Plot of Y values and Y estimates (optional)

Sample

The output listing for the sample problem is shown in Figure 15.

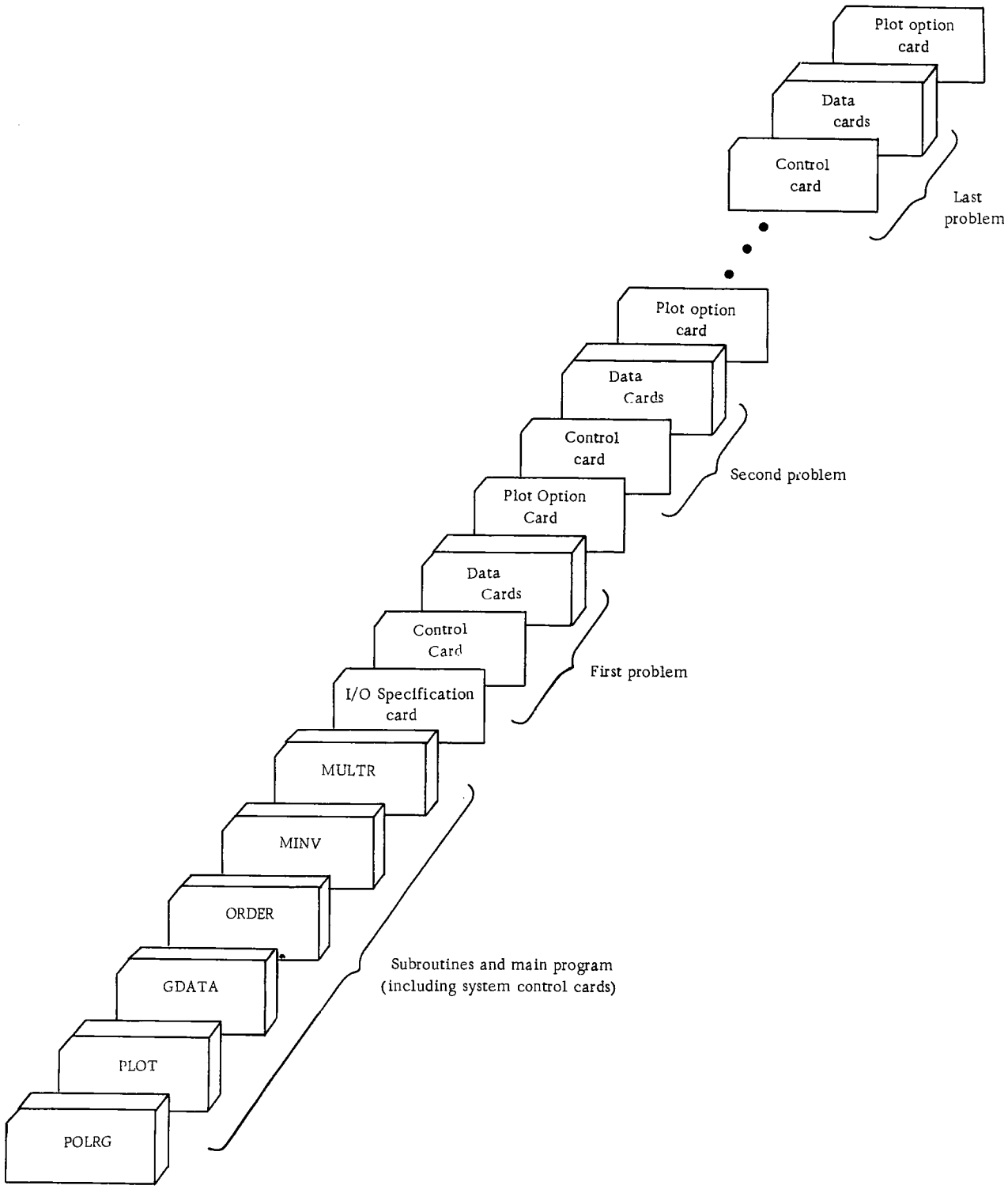


Figure 14. Deck setup (polynomial regression)

Program Modification

Noting that storage problems may result, as previously discussed in "Sample Program Description", program capacity can be increased or decreased by making changes in dimension statements. Input data in a different format can also be handled by providing a specific format statement. In order to familiarize the user with the program modification, the following general rules are supplied in terms of the sample problem:

1. Changes in the dimension statements of the main program, POLRG:
   a. The dimension of array X must be greater than or equal to the product of n (m + 1), where n is the number of observations and m is the highest degree polynomial to be fitted. Since there are 15 observations and the highest degree polynomial specified is 4, the product is 75 = 15 (4 + 1).
   b. The dimension of array DI must be greater than or equal to the product of m x m. For the sample problem this product is 16 = 4 x 4.
   c. The dimension of array D must be greater than or equal to (m + 2) (m + 1)/2. For the sample problem this number is 15 = (4 + 2) (4 + 1)/2.
   d. The dimensions of arrays B, E, SB, and T must be greater than or equal to the highest degree polynomial to be fitted, m. For the sample problem the value of m is 4.
   e. The dimension of arrays XBAR, STD, COE, SUMSQ and ISAVE must be greater than or equal to (m + 1). For the sample problem this value is 5 = (4 + 1).
   f. The dimension of array P must be greater than or equal to 3n. For the sample problem this value is 45 = 3(15). The array P is used when a plot of Y values and Y estimates is desired.
2. Changes in the input format statement of the main program, POLRG:

   Only the format statement for input data may be changed. Since sample data are either one-, two-, or three-digit numbers, rather than using six-column fields as in the sample problem, each row of data may be keypunched in two 3-column fields, and if so the format is changed to (2F 3. 0).

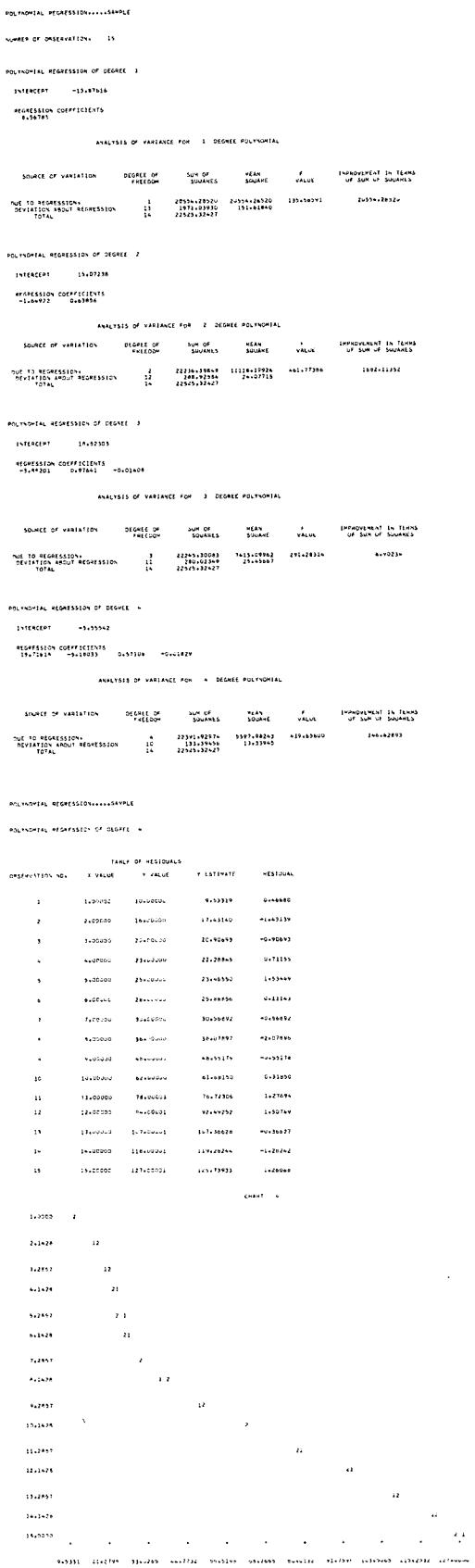


Figure 15. Output listing

Operating Instructions

The sample program for polynomial regression is a standard FORTRAN program. Special operating instructions are not required. Logical unit 2 is used for input, and logical unit 1 is used for output.

Sample Main Program for Polynomial Regression - POLRG

Purpose:

(1) Read the problem parameter card for a polynomial regression, (2) Call subroutines to perform the analysis, (3) Print the regression coefficients and analysis of variance table for polynomials of successively increasing degrees, and (4) Optionally print the table of residuals and a plot of Y values and Y estimates.

Remarks:

I/O specifications transmitted to subroutines by COMMON.

Input card:

Column 2 MX - Logical unit number for output.

Column 4 MY - Logical unit number for input.

The number of observations, N, must be greater than M+1, where M is the highest degree polynomial specified. If there is no reduction in the residual sum of squares between two successive degrees of the polynomials, the program terminates the problem before completing the analysis for the highest degree polynomial specified.

Subroutines and function subprograms required:

GDATA
ORDER
MINV
MULTR
PLOT (A special PLOT subroutine provided for the sample program.)

Method:

Refer to B. Ostle, 'Statistics in Research', The Iowa State College Press, 1954, chapter 6.

```
// FOR
*IOCS(CARD,TYPEWRITER,1132 PRINTER)
*ONE WORD INTEGERS
C       SAMPLE MAIN PROGRAM FOR POLYNOMIAL REGRESSION - POLRG                  POLRG  1
C       THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE           POLRG  2
C       PRODUCT OF N*(M+1), WHERE N IS THE NUMBER OF OBSERVATIONS AND          POLRG  3
C       M IS THE HIGHEST DEGREE POLYNOMIAL SPECIFIED.                          POLRG  4
      DIMENSION X(350)                                                         POLRGMO1
C       THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE           POLRG  6
C       PRODUCT OF M*M.                                                        POLRG  7
      DIMENSION DI(36)                                                         POLRGMO2
C       THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO               POLRG  9
C       (M+2)*(M+1)/2.                                                         POLRG 10
      DIMENSION DI(28)                                                         POLRGMO3
C       THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO M.           POLRG 12
      DIMENSION B(6),SB(6),T(6),E(6)                                           POLRGMO4
C       THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO              POLRG 14
C       (M+1).                                                                 POLRG 15
      DIMENSION XBAR(7),STD(7),COE(7),SUMSQ(7),ISAVE(7)                        POLRGMO5
C       THE FOLLOWIN DIMENSION MUST BE GREATER THAN OR EQUAL TO 10.            POLRG 17
      DIMENSION ANS(10)                                                        POLRG 18
C       THE FOLLOWING DIMENSION WILL BE USED IF THE PLOT OF OBSERVED           POLRG 19
C       DATA AND ESTIMATES IS DESIRED. THE SIZE OF THE DIMENSION, IN           POLRG 20
C       THIS CASE, MUST BE GREATER THAN OR EQUAL TO N*3. OTHERWISE,            POLRG 21
C       THE SIZE OF DIMENSION MAY BE SET TO 1.                                 POLRG 22
      DIMENSION P(150)                                                         POLRGMO6
      COMMON MX,MY                                                             POLRG 24
    1 FORMAT(A4,A2,I5,I2,I1)                                                   POLRG 25
    2 FORMAT(2F6.0)                                                            POLRG 26
    3 FORMAT(///27H POLYNOMIAL REGRESSION.....,A4,A2/)                         POLRG 27
    4 FORMAT(//23H NUMBER OF OBSERVATION,I6//)                                 POLRG 28
    5 FORMAT(//32H POLYNOMIAL REGRESSION OF DEGREE,I3)                         POLRG 29
    6 FORMAT(//12H    INTERCEPT,F15.5)                                         POLRG 30
    7 FORMAT(//26H    REGRESSION COEFFICIENTS/(10F12.5))                       POLRG 31
    8 FORMAT(///24X,24HANALYSIS OF VARIANCE FOR,I4,19H  DEGREE POLYNOMIAPOLRG 32
     1L/)                                                                      POLRG 33
    9 FORMAT(///,5X,19HSOURCE OF VARIATION,7X,9HDEGREE OF,7X,6HSUM OF,        POLRG 34
     19X,4HMEAN,10X,1HF,9X,20HIMPROVEMENT IN TERMS/33X,7HFREEDOM,8X,          POLRG 35
     27HSQUARES,7X,6HSQUARE,7X,5HVALUE,8X,17HOF SUM OF SQUARES)                POLRG 36
   10 FORMAT(//20H  DUE TO REGRESSION,12X,I6,F17.5,F14.5,F13.5,F20.5)          POLRG 37
   11 FORMAT(32H    DEVIATION ABOUT REGRESSION    ,I6,F17.5,F14.5)            POLRG 38
   12 FORMAT(8X,5HTOTAL,19X,I6,F17.5///)                                       POLRG 39
   13 FORMAT(//17H    NO IMPROVEMENT)                                         POLRG 40
   14 FORMAT(////27X,18HTABLE OF RESIDUALS//15H OBSERVATION NO.,5X,7HX VPOLRG 41
     1ALUE,7X,7HY VALUE,7X,10HY ESTIMATE,7X,8HRESIDUAL/)                       POLRG 42
   15 FORMAT(/,3X,I6,F18.5,F14.5,F17.5,F15.5)                                  POLRG 43
   16 FORMAT(2I2)                                                              POLRG 44
      READ(2,16)MX,MY                                                          POLRG 45
C       READ PROBLEM PARAMETER CARD                                            POLRG 46
  100 READ (MY,1) PR,PR1,N,M,NPLOT                                            POLRG 47
C       PR....PROBLEM NUMBER (MAY BE ALPHAMERIC)                               POLRG 48
C       PR1...PROBLEM NUMBER (CONTINUED)                                       POLRG 49
C       N.....NUMBER OF OBSERVATIONS                                           POLRG 50
C       M.....HIGHEST DEGREE POLYNOMIAL SPECIFIED                              POLRG 51
C       NPLOT,OPTION CODE FOR PLOTTING                                         POLRG 52
C             0  IF PLOT IS NOT DESIRED.                                       POLRG 53
C             1  IF PLOT IS DESIRED.                                           POLRG 54
C       PRINT PROBLEM NUMBER AND N.                                            POLRG 55
      WRITE (MX,3) PR,PR1                                                      POLRG 56
      WRITE (MX,4) N                                                           POLRG 57
C       READ INPUT DATA                                                        POLRG 58
      L=N*M                                                                    POLRG 59
      DO 110 I=1,N                                                             POLRG 60
      J=L+I                                                                    POLRG 61
C       X(I) IS THE INDEPENDENT VARIABLE, AND X(J) IS THE DEPENDENT            POLRG 62
C       VARIABLE.                                                              POLRG 63
  110 READ (MY,2) X(I),X(J)                                                    POLRG 64
      CALL GDATA (N,M,X,XBAR,STD,D,SUMSQ)                                      POLRG 65
      MM=M+1                                                                   POLRG 66
      SUM=0.0                                                                  POLRG 67
      NT=N-1                                                                   POLRG 68
      DO 200 I=1,M                                                             POLRG 69
      ISAVE(I)=I                                                               POLRG 70
C       FORM SUBSET OF CORRELATION COEFFICIENT MATRIX                          POLRG 71
      CALL ORDER (MM,D,MM,1,ISAVE,DI,E)                                        POLRG 72
C       INVERT THE SUBMATRIX OF CORRELATION COEFFICIENTS                       POLRG 73
      CALL MINV (DI,I,DET,B,T)                                                 POLRG 74
      CALL MULTR (N,I,XBAR,STD,SUMSQ,DI,E,ISAVE,B,SB,T,ANS)                    POLRG 75
C       PRINT THE RESULT OF CALCULATION                                        POLRG 76
      WRITE (MX,5) I                                                           POLRG 77
      IF(ANS(7))140,130,130                                                    POLRGMO7
  130 SUMIP=ANS(4)-SUM                                                         POLRGMO8
      IF(SUMIP) 140, 140, 150                                                  POLRG 79
  140 WRITE(MX,13)                                                             POLRG 80
      GO TO 210                                                                POLRG 81
  150 WRITE(MX,6)ANS(1)                                                        POLRG 82
      WRITE (MX,7) (B(J),J=1,I)                                                POLRG 83
      WRITE (MX,8) I                                                           POLRG 84
      WRITE (MX,9)                                                             POLRG 85
      SUM=ANS(4)                                                               POLRG 86
      WRITE (MX,10) I,ANS(4),ANS(6),ANS(10),SUMIP                              POLRG 87
      NI=ANS(8)                                                                POLRG 88
      WRITE (MX,11) NI,ANS(7),ANS(9)                                           POLRG 89
      WRITE (MX,12) NT,SUMSQ(MM)                                               POLRG 90
C       SAVE COEFFICIENTS FOR CALCULATION OF Y ESTIMATES                       POLRG 91
      COE(1)=ANS(1)                                                            POLRG 92
      DO 160 J=1,I                                                             POLRG 93
  160 COE(J+1)=B(J)                                                            POLRG 94
      LA=I                                                                     POLRG 95
  200 CONTINUE                                                                 POLRG 96
C       TEST WHETHER PLOT IS DESIRED                                           POLRG 97
  210 IF(NPLOT) 100, 100, 220                                                  POLRG 98
C       CALCULATE ESTIMATES                                                    POLRG 99
  220 NP3=N+N                                                                  POLRG100
      DO 230 I=1,N                                                             POLRG101
      NP3=NP3+1                                                                POLRG102
      P(NP3)=COE(1)                                                            POLRG103
      L=I                                                                      POLRG104
      DO 230 J=1,LA                                                            POLRG105
      P(NP3)=P(NP3)+X(L)*COE(J+1)                                              POLRG106
  230 L=L+N                                                                    POLRG107
C       COPY OBSERVED DATA                                                     POLRG108
      N2=N                                                                     POLRG109
       L=N*M                                                                   POLRG110
      DO 240 I=1,N                                                             POLRG111
      P(I)=X(I)                                                                POLRG112
      N2=N2+1                                                                  POLRG113
      L=L+1                                                                    POLRG114
  240 P(N2)=X(L)                                                               POLRG115
C       PRINT TABLE OF RESIDUALS                                               POLRG116
      WRITE (MX,3) PR,PR1                                                      POLRG117
      WRITE (MX,5) LA                                                          POLRG118
      WRITE (MX,14)                                                            POLRG119
      NP2=N                                                                    POLRG120
      NP3=N+N                                                                  POLRG121
      DO 250 I=1,N                                                             POLRG122
      NP2=NP2+1                                                                POLRG123
      NP3=NP3+1                                                                POLRG124
      RESID=P(NP2)-P(NP3)                                                      POLRG125
```

```
  250 WRITE (MX+15) I,P(1),P(NP2),P(NP3),RESID                    POLRG126
      CALL PLOT (LA,P,N,3,0.1)                                    POLRG127
      GO TO 100                                                   POLRG128
      END                                                         POLRG129
```

```
 1 2                                                                  1
SAMPLE00015041                                                        2
    1   10                                                            3
    2   16                                                            4
    3   20                                                            5
    4   23                                                            6
    5   25                                                            7
    6   26                                                            8
    7   30                                                            9
    8   36                                                           10
    9   48                                                           11
   10   62                                                           12
   11   78                                                           13
   12   94                                                           14
   13  107                                                           15
   14  118                                                           16
   15  127                                                           17
123456789                                                            18
```

```
      SUBROUTINE PLOT

      PURPOSE
         PLOT SEVERAL CROSS-VARIABLES VERSUS A BASE VARIABLE

      USAGE
         CALL PLOT (NO,A,N,M,NL,NS)

      DESCRIPTION OF PARAMETERS
         NO - CHART NUMBER (3 DIGITS MAXIMUM)
         A  - MATRIX OF DATA TO BE PLOTTED. FIRST COLUMN REPRESENTS
              BASE VARIABLE AND SUCCESSIVE COLUMNS ARE THE CROSS-
              VARIABLES (MAXIMUM IS 9).
         N  - NUMBER OF ROWS IN MATRIX A
         M  - NUMBER OF COLUMNS IN MATRIX A (EQUAL TO THE TOTAL
              NUMBER OF VARIABLES). MAXIMUM IS 10.
         NL - NUMBER OF LINES IN THE PLOT. IF 0 IS SPECIFIED, 50
              LINES ARE USED.
         NS - CODE FOR SORTING THE BASE VARIABLE DATA IN ASCENDING
              ORDER
                 0  SORTING IS NOT NECESSARY (ALREADY IN ASCENDING
                    ORDER).
                 1  SORTING IS NECESSARY.

      REMARKS
         NONE

      SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
         NONE
```

```
      SUBROUTINE PLOT(NO,A,N,M,NL,NS)                                 PLOT   1
      DIMENSION OUT(101),YPR(11),ANG(9),A(1)                          PLOT   2
      COMMON MX,MY                                                    PLOT   3
    1 FORMAT (///,60X,7H CHART ,I3,//)                                PLOT   4
    2 FORMAT (1X,F11.4,5X,101A1)                                      PLOT   5
    3 FORMAT (2X)                                                     PLOT   6
    5 FORMAT(10A1)                                                    PLOT   7
    7 FORMAT(    16X,101H.         .         .         .         .    PLOT   8
     1    .         .         .         .         .)                  PLOT   9
    8 FORMAT (//,9X,11F10.4)                                          PLOT  10
      NLL=NL                                                          PLOT  11
      IF(NS) 16, 16, 10                                               PLOT  12
C        SORT BASE VARIABLE DATA IN ASCENDING ORDER                   PLOT  13
   10 DO 15 I=1,N                                                     PLOT  14
      DO 14 J=1,N                                                     PLOT  15
      IF(A(I)-A(J)) 14, 14, 11                                        PLOT  16
   11 L=I-N                                                           PLOT  17
      LL=J-N                                                          PLOT  18
      DO 12 K=1,M                                                     PLOT  19
      L=L+N                                                           PLOT  20
      LL=LL+N                                                         PLOT  21
      F=A(L)                                                          PLOT  22
      A(L)=A(LL)                                                      PLOT  23
   12 A(LL)=F                                                         PLOT  24
   14 CONTINUE                                                        PLOT  25
   15 CONTINUE                                                        PLOT  26
C        TEST NLL                                                     PLOT  27
   16 IF(NLL) 20, 18, 20                                              PLOT  28
   18 NLL=50                                                          PLOT  29
C        PRINT TITLE                                                  PLOT  30
   20 WRITE(MX,1)NO                                                   PLOT  31
C        READ BLANK AND DIGITS FOR PRINTING                           PLOT  32
      READ(MY,5) BLANK,(ANG(I),I=1,9)                                 PLOT  33
C        FIND SCALE FOR BASE VARIABLE                                 PLOT  34
      XSCAL=(A(N)-A(1))/(FLOAT(NLL-1))                                PLOT  35
C        FIND SCALE FOR CROSS-VARIABLES                               PLOT  36
      M1=N+1                                                          PLOT  37
      M2=M*N                                                          PLOT  38
      YMIN=A(M1)                                                      PLOT  39
      YMAX=YMIN                                                       PLOT  40
      DO 40 J=M1,M2                                                   PLOT  41
      IF(A(J)-YMIN) 28,26,26                                          PLOT  42
   26 IF(A(J)-YMAX) 40,40,30                                          PLOT  43
   28 YMIN=A(J)                                                       PLOT  44
      GO TO 40                                                        PLOT  45
   30 YMAX=A(J)                                                       PLOT  46
   40 CONTINUE                                                        PLOT  47
      YSCAL=(YMAX-YMIN)/100.0                                         PLOT  48
C        FIND BASE VARIABLE PRINT POSITION                            PLOT  49
      XB=A(1)                                                         PLOT  50
      L=1                                                             PLOT  51
      MYX = M-1                                                       PLOT  52
      I=1                                                             PLOT  53
   45 F=I-1                                                           PLOT  54
      XPR=XB+F*XSCAL                                                  PLOT  55
      IF(A(L)-XPR) 50,50,70                                           PLOT  56
C        FIND CROSS-VARIABLES                                         PLOT  57
   50 DO 55 IX=1,101                                                  PLOT  58
   55 OUT(IX)=BLANK                                                   PLOT  59
      DO 60 J=1,MYX                                                   PLOT  60
      LL=L+J*N                                                        PLOT  61
      JP=((A(LL)-YMIN)/YSCAL)+1.0                                     PLOT  62
      OUT(JP)=ANG(J)                                                  PLOT  63
   60 CONTINUE                                                        PLOT  64
C        PRINT LINE AND CLEAR, OR SKIP                                PLOT  65
      WRITE(MX,2)XPR,(OUT(IZ),IZ=1,101)                               PLOT  66
      L=L+1                                                           PLOT  67
      GO TO 80                                                        PLOT  68
   70 WRITE(MX,3)                                                     PLOT  69
   80 I=I+1                                                           PLOT  70
      IF(I-NLL)45,84,85                                               PLOT  71
   84 XPR=A(N)                                                        PLOT  72
      GO TO 50                                                        PLOT  73
C        PRINT CROSS-VARIABLES NUMBERS                                PLOT  74
   86 WRITE(MX,7)                                                     PLOT  75
      YPR(1)=YMIN                                                     PLOT  76
      DO 90 KN=1,9                                                    PLOT  77
   90 YPR(KN+1)=YPR(KN)+YSCAL*10.0                                    PLOT  78
      YPR(11)=YMAX                                                    PLOT  79
      WRITE(MX,8)(YPR(IP),IP=1,11)                                    PLOT  80
      RETURN                                                          PLOT  81
      END                                                             PLOT  82
```

CANONICAL CORRELATION

Problem Description

An analysis of the interrelations between two sets of variables measured on the same subjects is performed by this program. These variables are predictors in one set and criteria in the other set, but it is irrelevant whether the variables in the first set or in the second set are considered as the prediction variables. The canonical correlation, which gives the maximum correlation between linear functions of the two sets of variables, is calculated. χ^2 is also computed to test the significance of canonical correlation.

The sample problem for canonical correlation consists of four variables in the first set (left-hand side) and three variables in the second set (right-hand side) as presented in Table 4. These two sets of measurements have been made on 23 subjects.

Table 4. Sample Data for Canonical Correlation

	First set				Second set		
Observation	X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3
1	191	155	65	19	179	145	70
2	195	149	70	20	201	152	69
3	181	148	71	19	185	149	75
4	183	153	82	18	188	149	86
5	176	144	67	18	171	142	71
6	208	157	81	22	192	152	77
7	189	150	75	21	190	149	72
8	197	159	90	20	189	152	82
9	188	152	76	19	197	159	84
10	192	150	78	20	187	151	72
11	179	158	99	18	186	148	89
12	183	147	65	18	174	147	70
13	174	150	71	19	185	152	65
14	190	159	91	19	195	157	99
15	188	151	98	20	187	158	87
16	163	137	59	18	161	130	63
17	195	155	85	20	183	158	81
18	196	153	80	21	173	148	74
19	181	145	77	20	182	146	70
20	175	140	70	19	165	137	81
21	192	154	69	20	185	152	63
22	174	143	79	20	178	147	73
23	176	139	70	20	176	143	69

Program

Description

The canonical correlation sample program consists of a main routine, MCANO, and six subroutines:

CORRE CANOR MINV NROOT EIGEN	are from the Scientific Subroutine Package
DATA	is a special input subroutine

Capacity

The capacity of the sample program and the format required for data input have been set up as follows:

1. Up to 9 variables, including both the first set of variables (that is, left-hand variables) and the second set of variables (that is, right-hand variables). The number of variables in the first set must be greater than or equal to the number of variables in the second set.
2. Up to 99,999 observations.
3. (12F 6.0) format for input data cards.

Therefore, if a problem satisfies the above conditions it is not necessary to modify the sample program. However, if there are more than 9 variables, dimension statements in the sample main program must be modified to handle the particular problem. Similarly, if input data cards are prepared using a different format, the input format in the input subroutine, DATA, must be modified. The general rules for program modification are described later.

Input

I/O Specification Card

One control card is required for each problem and is read by the main program, MCANO. This card is prepared as follows:

Columns	Contents	For Sample Problem
1 - 6	Problem number (may be alphameric)	SAMPLE
7 - 11	Number of observations	00023
12 - 13	Number of variables in the first set (that is, left-hand variables)*	04
14 - 15	Number of variables in the second set (that is, right-hand variables)	03

*The number of variables in the first set must be greater than or equal to the number of variables in the second set.

Leading zeros are not required to be keypunched; but must be right-justified within fields.

Data Cards

Since input data are read into the computer one observation at a time, each row of data in Table 4 is keypunched on a separate card using the format (12F 6.0). This format assumes twelve 6-column fields per card.

Deck Setup

Deck setup is shown in Figure 16.

Sample

The listing of input cards for the sample problem is presented at the end of the sample main program.

Output

Description

The output of the sample program for canonical correlation includes:

1. Means
2. Standard deviations
3. Correlation coefficients
4. Eigenvalues and corresponding canonical correlation
5. Lambda
6. Chi-square and degrees of freedom
7. Coefficients for left- and right-hand variables

Sample

The output listing for the sample problem is shown in Figure 17 of this sample program.

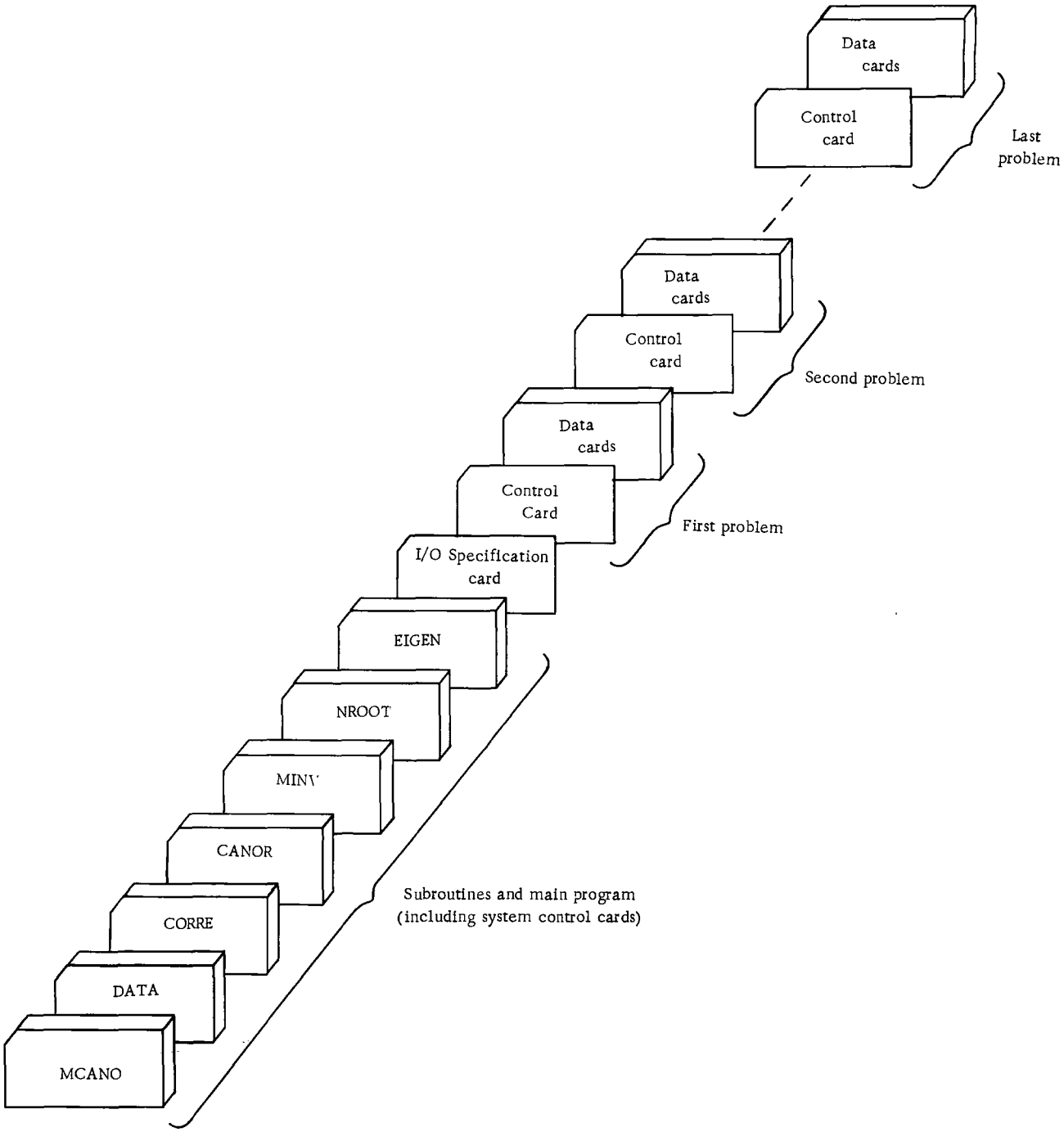
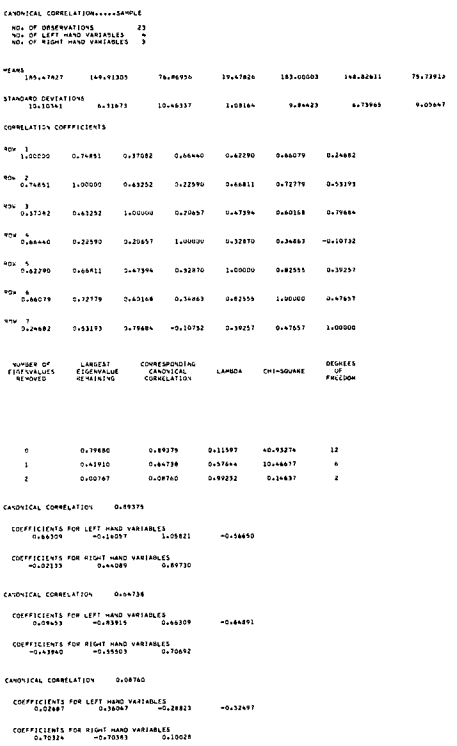


Figure 16. Deck setup (canonical correlation)

Program Modification

Noting that storage problems may result, as previously described in "Sample Program Descriptions", program capacity can be increased or decreased by making changes in dimension statements. Input data in a different format can also be handled by providing a specific format statement. In order to familiarize



```
CANONICAL CORRELATION.....SAMPLE
 NO. OF OBSERVATIONS          23
 NO. OF LEFT HAND VARIABLES    4
 NO. OF RIGHT HAND VARIABLES   3

MEANS
 145.47427     149.91305     76.86956     19.47826     183.00000     148.82611     75.73913

STANDARD DEVIATIONS
 10.10341      8.31673      10.46337      1.08164      9.84423       8.73985      9.03647

CORRELATION COEFFICIENTS

ROW 1
 1.00000      0.74851      0.37082      0.66440      0.62290      0.86079      0.24882

ROW 2
 0.74851      1.00000      0.63252      0.22590      0.66811      0.72779      0.53193

ROW 3
 0.37082      0.63252      1.00000      0.20657      0.47394      0.60168      0.79684

ROW 4
 0.66440      0.22590      0.20657      1.00000      0.32870      0.34863     -0.10732

ROW 5
 0.62290      0.66811      0.47394      0.32870      1.00000      0.82555      0.39257

ROW 6
 0.86079      0.72779      0.60168      0.34863      0.82555      1.00000      0.47657

ROW 7
 0.24882      0.53193      0.79684     -0.10732      0.39257      0.47657      1.00000

 NUMBER OF    LARGEST      CORRESPONDING
EIGENVALUES  EIGENVALUE     CANONICAL      LAMBDA     CHI-SQUARE   DEGREES
REMOVED     REMAINING     CORRELATION                             OF
                                                                  FREEDOM

    0          0.79686       0.89379       0.11597     40.93274      12
    1          0.43910       0.66738       0.57844     10.44677       6
    2          0.00767       0.08760       0.99232      0.16437       2

CANONICAL CORRELATION=    0.89379

 COEFFICIENTS FOR LEFT HAND VARIABLES
    0.86309      -0.16097       1.05821      -0.56650

 COEFFICIENTS FOR RIGHT HAND VARIABLES
   -0.02133       0.44089       0.89730

CANONICAL CORRELATION=    0.66736

 COEFFICIENTS FOR LEFT HAND VARIABLES
    0.09453      -0.85915       0.66309      -0.84891

 COEFFICIENTS FOR RIGHT HAND VARIABLES
   -0.43940      -0.55503       0.70692

CANONICAL CORRELATION=    0.08760

 COEFFICIENTS FOR LEFT HAND VARIABLES
    0.02887       0.36047      -0.28823      -0.32497

 COEFFICIENTS FOR RIGHT HAND VARIABLES
    0.70324      -0.70383       0.10028
```

Figure 17. Output listing

the user with the program modification, the following general rules are supplied in terms of the sample problem:

1. Changes in the dimension statements of the main program, MCANO:
   a. The dimension of arrays XBAR, STD, CANR, CHISQ, and NDF must be greater than or equal to the total number of variables m (m = p + q, where p is the number of left-hand variables and q is the number of right-hand variables). Since there are seven variables, four on left and three on right, the value of m is 7.
   b. The dimension of array RX must be greater than or equal to the product of m x m. For the sample problem this product is 49 = 7 x 7.
   c. The dimension of array R must be greater than or equal to (m + 1)m/2. For the sample problem this number is 28 = (7 + 1)7/2.
   d. The dimension of array COEFL must be greater than or equal to the product of p x q. For the sample problem this product is 12 = 4 x 3.
   e. The dimension of array COEFR must be greater than or equal to the product of q x q. For the sample problem this product is 9 = 3 x 3.

2. Changes in the input format statement of the special input subroutine, DATA:
   Only the format statement for input data may be changed. For example, since sample data are either two- or three-digit numbers, rather than using six-column fields as in the sample problem, each row of data may be keypunched in seven 3-column fields, and if so, the format would be changed to (7F 3.0). Note that the current input format statement will allow a maximum of twelve variables per card. The special input subroutine, DATA, is normally written by the user to handle different formats for different problems. The user may modify this subroutine to perform testing of input data, transformation of data, and so on.

Operating Instructions

The sample program for canonical correlation is a standard FORTRAN program. Special operating instructions are not required. Logical unit 2 is used for input, and logical unit 1 is used for output.

Sample Main Program for Canonical Correlation - MCANO

Purpose:

(1) Read the problem parameter card for a canonical correlation, (2) Call two subroutines to calculate simple correlations, canonical correlations, chi-squares, degrees of freedom for chi-squares, and coefficients for left and right hand variables, namely canonical variates, and (3) Print the results.

Remarks:

I/O specifications transmitted to subroutines by COMMON.

Input card:

Column 2 MX - Logical unit number for output.

Column 4 MY - Logical unit number for input.

The number of left-hand variables must be greater than or equal to the number of right-hand variables.

Subroutines and function subprograms required:

CORRE (which, in turn, calls the input subroutine named DATA.)

CANOR (which, in turn, calls the subroutines MINV and NROOT. NROOT, in turn, calls the subroutine EIGEN.)

Method:

Refer to W. W. Cooley and P. R. Lohnes, 'Multivariate Procedures for the Behavioral Sciences', John Wiley and Sons, 1962, chapter 3.

```
// FOR
*IOCS(CARD,TYPEWRITER,1132 PRINTER)
*ONE WORD INTEGERS
C         SAMPLE MAIN PROGRAM FOR CANONICAL CORRELATION - MCANO          MCANO  1
C         THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO THE  MCANO  2
C         TOTAL NUMBER OF VARIABLES M (M=MP+MQ, WHERE MP IS THE NUMBER   MCANO  3
C         OF LEFT HAND VARIABLES, AND MQ IS THE NUMBER OF RIGHT HAND     MCANO  4
C         VARIABLES).                                                    MCANO  5
      DIMENSION XBAR(9),STD(9),CANR(9),CHISQ(9),NDF(9)                   MCANO  6
C         THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE   MCANO  7
C         PRODUCT OF M*M.                                                MCANO  8
      DIMENSION RX(81)                                                   MCANO  9
C         THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO       MCANO 10
C         (M+1)*M/2.                                                     MCANO 11
      DIMENSION R(45)                                                    MCANO 12
C         THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE   MCANO 13
C         PRODUCT OF MP*MQ.                                              MCANO 14
      DIMENSION COEFL(81)                                                MCANO 15
C         THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE   MCANO 16
C         PRODUCT OF MQ*MQ.                                              MCANO 17
      DIMENSION COEFR(25)                                                MCANO 18
        COMMON MX,MY                                                     MCANO 19
    1 FORMAT(A4,A2,I5,2I2)                                               MCANO 20
    2 FORMAT(////27H CANONICAL CORRELATION.....,A4,A2//22H   NO. OF OBSEMCANO 21
     1RVATIONS,8X,I4/29H   NO. OF LEFT HAND VARIABLES,I5/30H   NO. OF RIMCANO 22
     2GHT HAND VARIABLES,I4/)                                            MCANO 23
    3 FORMAT(//6H MEANS/(8F15.5))                                        MCANO 24
    4 FORMAT(//20H STANDARD DEVIATIONS/(8F15.5))                         MCANO 25
    5 FORMAT(//25H CORRELATION COEFFICIENTS)                             MCANO 26
    6 FORMAT(//4H ROW,I3/(10F12.5))                                      MCANO 27
    7 FORMAT(////12H   NUMBER OF,7X,7HLARGEST,7X,13HCORRESPONDING,31X,7HMCANO 28
     1DEGREES/13H  EIGENVALUES,5X,10HEIGENVALUE,7X,9HCANONICAL,7X,6HLAMBMCANO 29
     20A,5X,10HCHI-SQUARE,7X,2HOF/4X,7HREMOVED,7X,9HREMAINING,7X,11HCORRMCANO 30
     3ELATION,32X,7HFREEDOM/)                                            MCANO 31
    8 FORMAT(/17,F19.5,F16.5,2F14.5,5X,I5)                               MCANO 32
    9 FORMAT(///22H CANONICAL CORRELATION,F12.5)                         MCANO 33
   10 FORMAT(//39H   COEFFICIENTS FOR LEFT HAND VARIABLES/(8F15.5))      MCANO 34
   11 FORMAT(//40H   COEFFICIENTS FOR RIGHT HAND VARIABLES/(8F15.5))     MCANO 35
   12 FORMAT(2I2)                                                        MCANO 36
      READ(2,12)MX,MY                                                    MCANO 37
C         READ PROBLEM PARAMETER CARD                                    MCANO 38
  100 READ (MY,1)PR,PR1,N,MP,MQ                                         MCANO 39
C         PR.......PROBLEM NUMBER (MAY BE ALPHAMERIC)                    MCANO 40
C         PR1......PROBLEM NUMBER (CONTINUED)                            MCANO 41
C         N........NUMBER OF OBSERVATIONS                                MCANO 42
C         MP.......NUMBER OF LEFT HAND VARIABLES                         MCANO 43
C         MQ.......NUMBER OF RIGHT HAND VARIABLES                        MCANO 44
      WRITE (MX,2)PR,PR1,N,MP,MQ                                         MCANO 45
      M=MP+MQ                                                            MCANO 46
      IO=0                                                               MCANO 47
      X=0.0                                                              MCANO 48
      CALL CORRE (N,M,IO,X,XBAR,STD,RX,R,CANR,CHISQ,COEFL)               MCANO 49
C         PRINT MEANS, STANDARD DEVIATIONS, AND CORRELATION              MCANO 50
C         COEFFICIENTS OF ALL VARIABLES                                  MCANO 51
      WRITE (MX,3)(XBAR(I),I=1,M)                                        MCANO 52
      WRITE (MX,4)(STD(I),I=1,M)                                         MCANO 53
      WRITE (MX,5)                                                       MCANO 54
      DO 160 I=1,M                                                       MCANO 55
      DO 150 J=1,M                                                       MCANO 56
      IF(I-J) 120, 130, 130                                              MCANO 57
  120 L=I+(J*J-J)/2                                                      MCANO 58
      GO TO 140                                                          MCANO 59
  130 L=J+(I*I-I)/2                                                      MCANO 60
  140 CANR(J)=R(L)                                                       MCANO 61
  150 CONTINUE                                                           MCANO 62
  160 WRITE (MX,6)I,(CANR(J),J=1,M)                                      MCANO 63
      CALL CANOR (N,MP,MQ,R,XBAR,STD,CANR,CHISQ,NDF,COEFR,COEFL,RX)      MCANO 64
C         PRINT EIGENVALUES, CANONICAL CORRELATIONS, LAMBDA, CHI-SQUARES MCANO 65
C         DEGREES OF FREEDOMS                                            MCANO 66
      WRITE (MX,7)                                                       MCANO 67
      DO 170 I=1,MQ                                                      MCANO 68
      N1=I-1                                                             MCANO 69
C         TEST WHETHER EIGENVALUE IS GREATER THAN ZERO                   MCANO 70
      IF(XBAR(I)) 165, 165, 170                                          MCANO 71
  165 MM=N1                                                              MCANO 72
      GO TO 175                                                          MCANO 73
  170 WRITE (MX,8)N1,XBAR(I),CANR(I),STD(I),CHISQ(I),NDF(I)              MCANO 74
      MM=MQ                                                              MCANO 75
C         PRINT CANONICAL COEFFICIENTS                                   MCANO 76
  175 N1=0                                                               MCANO 77
      N2=0                                                               MCANO 78
      DO 200 I=1,MM                                                      MCANO 79
      WRITE (MX,9)CANR(I)                                                MCANO 80
      DO 180 J=1,MP                                                      MCANO 81
      N1=N1+1                                                            MCANO 82
  180 XBAR(J)=COEFL(N1)                                                  MCANO 83
      WRITE (MX,10)(XBAR(J),J=1,MP)                                      MCANO 84
      DO 190 J=1,MQ                                                      MCANO 85
      N2=N2+1                                                            MCANO 86
  190 XBAR(J)=COEFR(N2)                                                  MCANO 87
      WRITE (MX,11)(XBAR(J),J=1,MQ)                                      MCANO 88
  200 CONTINUE                                                           MCANO 89
      GO TO 100                                                          MCANO 90
      END                                                                MCANO 91
// DUP
*STORE      WS  UA  MCANO
// XEQ MCANO   01
*LOCALMCANO,CORRE,CANOR
```

```
 1 2                                                                     1
SAMPLE000230403                                                          2
  191   155    65    19   179   145    70                               3
  195   149    70    20   201   152    69                               4
  181   148    71    19   185   149    75                               5
  183   153    82    18   188   149    86                               6
  176   144    67    18   171   142    71                               7
  208   157    81    22   192   152    77                               8
  189   150    75    21   190   149    72                               9
  197   159    90    20   189   152    82                              10
  188   152    76    19   197   159    84                              11
  192   150    78    20   187   151    72                              12
  179   158    99    18   186   148    89                              13
  183   147    65    18   174   147    70                              14
  174   150    71    19   185   152    65                              15
  190   159    91    19   195   157    99                              16
  188   151    98    20   187   158    87                              17
  163   137    59    18   161   130    63                              18
  195   155    85    20   183   158    81                              19
  196   153    80    21   173   148    76                              20
  181   145    77    20   182   146    70                              21
  175   140    70    19   165   137    81                              22
  192   154    69    20   185   152    63                              23
  174   143    79    20   178   147    73                              24
  176   139    70    20   176   143    69                              25
```

SAMPLE INPUT SUBROUTINE - DATA

PURPOSE
READ AN OBSERVATION (N DATA VALUES) FROM INPUT DEVICE. THIS SUBROUTINE IS CALLED BY THE SUBROUTINE CORRE AND MUST BE PROVIDED BY THE USER. IF SIZE AND LOCATION OF DATA FIELDS ARE DIFFERENT FROM PROBLEM TO PROBLEM, THIS SUBROUTINE MUST BE RECOMPILED WITH A PROPER FORMAT STATEMENT.

USAGE
CALL DATA (N,D)

DESCRIPTION OF PARAMETERS
N - THE NUMBER OF VARIABLES IN AN OBSERVATION.
D - OUTPUT VECTOR OF LENGTH N CONTAINING THE OBSERVATION DATA.

REMARKS
THE TYPE OF CONVERSION SPECIFIED IN THE FORMAT MUST BE EITHER F OR E.

SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
NONE

```
      SUBROUTINE DATA (M,D)                                              DATA   1
      DIMENSION D(1)                                                     DATA   2
      COMMON MX,MY                                                       DATA   3
    1 FORMAT(12F6.0)                                                     DATA   4
C         READ AN OBSERVATION FROM INPUT DEVICE.                         DATA   5
      READ (MY,1) (D(I),I=1,M)                                           DATA   6
      RETURN                                                             DATA   7
      END                                                                DATA   8
```

ANALYSIS OF VARIANCE

Problem Description

An analysis of variance is performed for a factorial design by use of three special operators suggested by H.O. Hartley.* The analysis of many other designs can be derived by reducing them first to factorial designs, and then pooling certain components of the analysis-of-variance table.

Consider a three-factor factorial experiment in a randomized complete block design as present in Table 5. In this experiment factor A has four levels, factors B and C have three levels, and the entire experiment is replicated twice. The replicates are completely unrelated and do not constitute a factor.

Table 5. Sample Data for Analysis of Variance

Replicate (Block)		b_1				b_2				b_3			
		a_1	a_2	a_3	a_4	a_1	a_2	a_3	a_4	a_1	a_2	a_3	a_4
r_1....	c_1	3	10	9	8	24	8	9	3	2	8	9	8
	c_2	4	12	3	9	22	7	16	2	2	2	7	2
	c_3	5	10	5	8	23	9	17	3	2	8	6	3
r_2....	c_1	2	14	9	13	29	16	11	3	2	7	5	3
	c_2	7	11	5	8	28	18	10	6	6	6	5	9
	c_3	9	10	27	8	28	16	11	7	8	9	8	15

*H.O. Hartley, "Analysis of Variance" in Mathematical Methods for Digital Computers, edited by A. Ralston and H. Wilf, John Wiley and Sons, 1962, Chapter 20.

Nevertheless, for the purpose of this program, a four-factor experiment (with factors A, B, C, and R) is assumed. Thus, each element of the data in Table 5 may be represented in the form:

$$x_{abcr} \quad \text{where} \quad a = 1,2,3,4$$
$$b = 1,2,3$$
$$c = 1,2,3$$
$$r = 1,2$$

The general principle of the analysis-of-variance procedure used in the program is to perform first a formal factorial analysis and then pool certain components in accordance with summary instructions that specifically apply to the particular design. The summary instructions for four different designs are presented in the output section.

Program

Description

The analysis-of-variance sample program consists of a main routine, ANOVA, and three subroutines:

AVDAT
AVCAL
MEANQ

} are from the Scientific Subroutine Package

Capacity

The capacity of the sample program and the format required for data input have been set up as follows:

1. Up to six-factor factorial experiment
2. Up to a total of 1600 data points. The total number of core locations for data points in a problem is calculated as follows:

$$T = \prod_{i=1}^{k} (LEVEL_i + 1)$$

where $LEVEL_i$ = number of levels of i^{th} factor
k = number of factors
$\prod$ = notation for repeated products

3. (12F6.0) format for input data cards

Therefore, if a problem satisfies the above conditions it is not necessary to modify the sample program. However, if there are more than six factors or if the total number of data points is more than 1800, dimension statements in the sample main program must be modified. Similarly, if input data cards are prepared using a different format, the input format statement in the sample main program must be modified. The general rules for program modifications are described later.

Input

I/O Specification Card

One control card is required for each problem and is read by the main program, ANOVA. This card is prepared as follows:

Columns	Contents	For Sample Problem
1 - 6	Problem number (may be alphameric)	SAMPLE
7 - 8	Number of factors	04
9 - 15	Blank	
16	Label for the first factor	A
17 - 20	Number of levels of the first factor	0004
21	Label for the second factor	B
22 - 25	Number of levels of the second factor	0003
	⋮	
26	Label for the third factor	C
27 - 30	Number of levels of the third factor	0003
31	Label for the fourth factor	R
32 - 35	Number of levels of the fourth factor	0002
66	Label for the eleventh factor (if present)	
67 - 70	Number of levels of the eleventh factor	

If there are more than eleven factors, continue to the second card in the same manner.

Columns	Contents
1	Label for the twelfth factor
2 - 5	Number of levels for the twelfth factor
.	
.	
.	
etc.	

Leading zeros are not required to be keypunched.

Data Cards

Data are keypunched in the following order: X_{1111}, X_{2111}, X_{3111}, X_{4111}, X_{1211}, X_{2211}, X_{3211},..., X_{4332}. In other words, the innermost subscript is changed first; namely, the first factor, and then second, third, and fourth subscripts. In the sample problem, the first subscript corresponds to factor A and the second, third, and fourth subscripts to factors B, C, and R. Since the number of data fields per cards is twelve, implied by the format (12F6.0), each row in Table 5 is keypunched on a separate card.

Deck Setup

Deck setup is shown in Figure 18.

Sample

The listing of input cards for the sample problem is presented at the end of the sample main program.

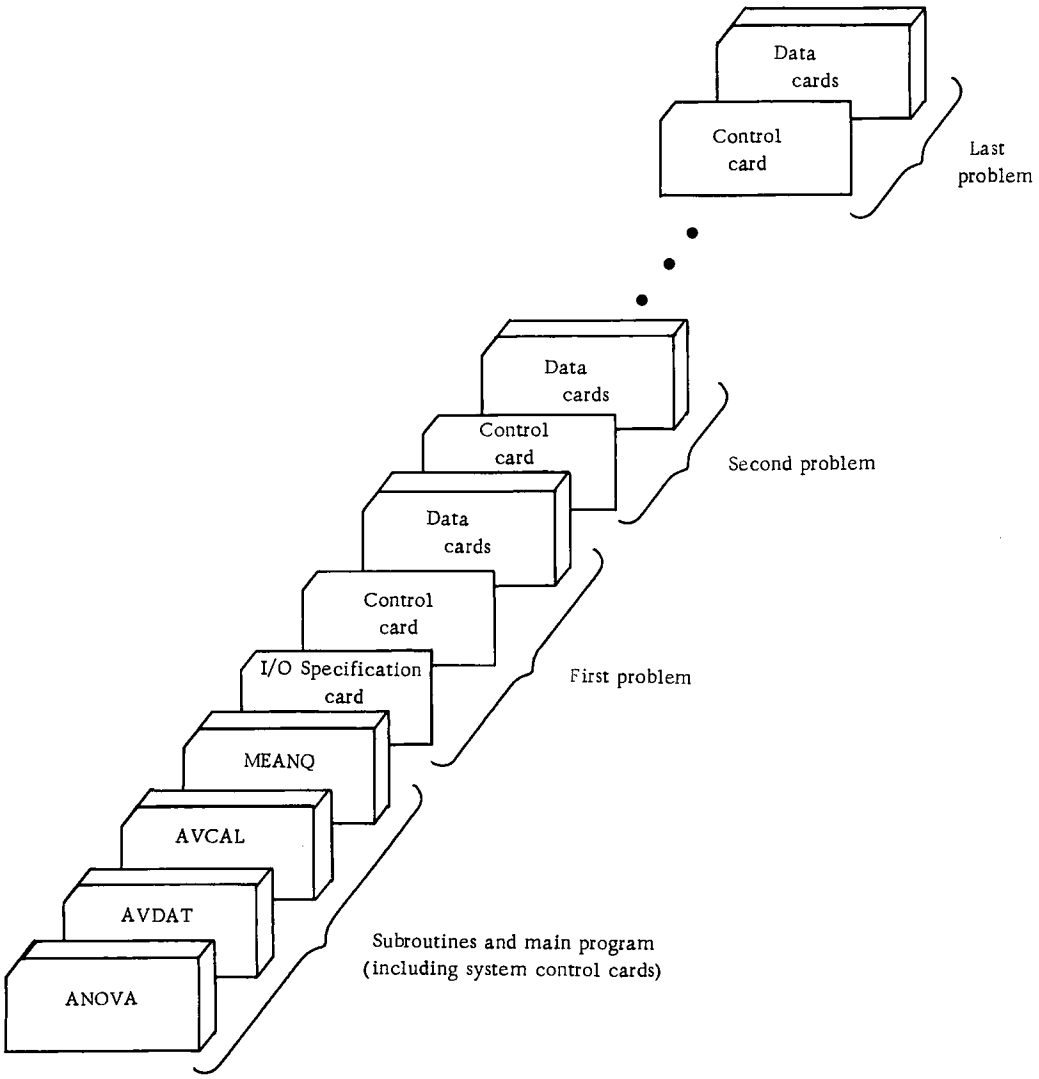


Figure 18. Deck setup (analysis of variance)

Output

Description

The output of the sample analysis-of-variance program includes the numbers of levels of factors as input, the mean of all data, and the table of analysis of variance. In order to complete the analysis of variance properly, however, certain components in the table may need to be pooled. This is accomplished by means of summary instructions that specifically apply to the particular experiment as presented in Table 6.

Table 6. Instructions to Summarize Components of Analysis of Variance

	Single Classification with Replicates	Two-way Classification with Cell Replicates	Randomized Complete Block with Two Factors	Split Plot
(Input) Factor No. 1 2 3	Groups = A Replicates = R	Rows = A Columns = B Replicates = R	Factor 1 = A Factor 2 = B Blocks = R	Main treatment = A Subtreatment = B Blocks = R
(Output) Sums of squares	A R AR	A B AB R AR BR ABR	A B AB R AR BR ABR	A B AB R AR BR ABR
Summary instruction	Error = R + (AR)	Error = R + (AR) +(BR)+(ABR)	Error = (AR)+(BR) +(ABR)	Error = (BR)+(ABR) (b)
Analysis of variance	Groups A Error	Rows A Columns B Interaction AB Error	Factor 1 A Factor 2 B Interaction AB Blocks R Error	Main treatment A Blocks R Error (a) AR Subtreatment B Interaction AB Error (b)

As mentioned earlier, the sample problem is a randomized complete block design with three factors replicated twice. Therefore, it is necessary to pool certain components in the table of analysis of variance shown in Figure 19. Specifically, the components AR, BR, ABR, CR, ACR, BCR, and ABCR are combined into one value called the error term. The result is indicated in Figure 19. Since these data are purely hypothetical, interpretations of the various effects are not made.

Sample

The output listing for the sample problem is shown in Figure 19.

```
ANALYSIS OF VARIANCE.....SAMPLE


LEVELS OF FACTORS
  A          4
  B          3
  C          3
  R          2


GRAND MEAN              9.40277


SOURCE OF VARIATION     SUMS OF SQUARES     DEGREES OF FREEDOM     MEAN SQUARES
A                        229.04168                   3              76.34722
B                        722.69445                   2             361.34722
AB                      1382.08349                   6             230.34722
C                         55.11111                   2              27.55555
AC                        42.00000                   6               7.00000
BC                        13.13888                   4               3.28472
ABC                      140.75003                  12              11.72916
R                        141.68057                   1             141.68057
AR                        18.81944                   3               6.27314
BR                         6.02777                   2               3.01388
ABR                      176.97222                   6              29.49536
CR                        40.77777                   2              20.38888
ACR                       50.55555                   6               8.42592
BCR                       62.63889                   4              15.65972
ABCR                     151.02780                  12              12.58564

TOTAL                   3233.31641                  71
```

Figure 19. Output listing

Program Modification

Noting that storage problems may result, as previously described in "Sample Program Description", program capacity can be increased or decreased by making changes in dimension statements. Input data in a different format can also be handled by providing a specific format statement. In order to familiarize the user with the program modification the following general rules are supplied in terms of the sample problem:

1. Changes in the dimension statements of the main program, ANOVA:
   a. The dimension of array X must be greater than or equal to the total number of data points as calculated by the formula in the program capacity section above. For the sample problem the total number of data points is 240 = (4+1)(3+1)(3+1)(2+1).
   b. The dimension of arrays HEAD, LEVEL, ISTEP, KOUNT, and LASTS must be greater than or equal to the number of factors, k. Since there are four factors in the sample problem (4 = 3 original factors + 1 pseudo factor) the value of k is 4.
   c. The dimension of arrays SUMSQ, NDF, and SMEAN must be greater than or equal to $n = 2^k-1$, where k is the number of factors. For the sample problem the value of n is $15 = 2^4-1$.
2. Change in the input format statement of the main program, ANOVA:
   Only the format statement for input data may be changed. Since sample data are either one- or two-digit numbers, *rather than using*

six-column fields as in the sample problem, each data may be keypunched in a two-column field, and, if so, the format is changed to (12F2.0). This format assumes twelve 2-column fields per card, beginning in column 1.

Operating Instructions

The sample analysis-of-variance program is a standard FORTRAN program. Special operating instructions are not required. Logical unit 2 is used for input, and logical unit 1 is used for output.

Sample Main Program for Analysis of Variance - ANOVA

Purpose:

(1) Read the problem parameter card for analysis of variance, (2) Call the subroutines for the calculation of sums of squares, degrees of freedom and mean square, and (3) Print factor levels, grand mean, and analysis of variance table.

Remarks:

The program handles only complete factorial designs. Therefore, other experimental design must be reduced to this form prior to the use of the program.
I/O logical units determined by MX and MY, respectively.

Subroutines and function subprograms required:

AVDAT
AVCAL
MEANQ

Method:

The method is based on the technique discussed by H.O. Hartley in "Mathematical Methods for Digital Computers", edited by A. Ralston and H. Wilf, John Wiley and Sons, 1962, Chapter 20.

```
// FOR
*IOCS(CARD,TYPEWRITER,1132 PRINTER)
*ONE WORD INTEGERS
C         SAMPLE MAIN PROGRAM FOR ANALYSIS OF VARIANCE - ANOVA                ANOVA  1
C         THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE        ANOVA  2
C         CUMULATIVE PRODUCT OF EACH FACTOR LEVEL PLUS ONE (LEVEL(I)+1)       ANOVA  3
C         FOR I=1 TO K, WHERE K IS THE NUMBER OF FACTORS..                    ANOVA  4
      DIMENSION X(1600)                                                       ANOVAM02
C         THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO THE       ANOVA  6
C         NUMBER OF FACTORS..                                                 ANOVA  7
      DIMENSION HEAD(6),LEVEL(6),ISTEP(6),KOUNT(6),LASTS(6)                   ANOVA  8
C         THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO 2 TO      ANOVA  9
C         THE K-TH POWER MINUS 1, ((2**K)-1)..                                ANOVA 10
      DIMENSION SUMSQ(63),NDF(63),SMEAN(63)                                   ANOVAM01
C         THE FOLLOWING DIMENSION IS USED TO PRINT FACTOR LABELS IN           ANOVA 12
C         ANALYSIS OF VARIANCE TABLE AND IS FIXED                             ANOVA 13
      DIMENSION FMT(15)                                                       ANOVA 14
    1 FORMAT(A4,A2,I2,A4,3X,11(A1,I4)/(A1,I4,A1,I4,A1,I4,A1,I4,A1,I4))       ANOVA 15
    2 FORMAT(////26H ANALYSIS OF VARIANCE.....A4,A2//)                        ANOVA 16
    3 FORMAT(//18H LEVELS OF FACTORS/(3X,A1,7X,I4))                           ANOVA 17
    4 FORMAT(////11H GRAND MEAN,F20.5////)                                    ANOVA 18
    5 FORMAT(//16H SOURCE OF,18X,7HSUMS OF,10X,10HDEGREES OF,9X,4HMEAN/       ANOVA 19
     110H VARIATION,18X,7HSQUARES,11X,7HFREEDOM,10X,7HSQUARES/)               ANOVA 20
    6 FORMAT(2X,15A1,F20.5,10X,I6,F20.5)                                      ANOVA 21
    7 FORMAT(/7H  TOTAL,10X,F20.5,10X,I6)                                     ANOVA 22
    8 FORMAT(12F6.0)                                                          ANOVA 23
    9 FORMAT(2I2)                                                             ANOVA 24
C         ..................................................................ANOVA 25
      READ(2,9)MX,MY                                                          ANOVA 26
C         READ PROBLEM PARAMETER CARD                                         ANOVA 27
  100 READ (MY,1)PR,PR1,K,BLANK,(HEAD(I),LEVEL(I),I=1,K)                     ANOVA 28
C           PR.....PROBLEM NUMBER (MAY BE ALPHAMERIC)                         ANOVA 29
C           PR1....PROBLEM NUMBER (CONTINUED)                                 ANOVA 30
C           K......NUMBER OF FACTORS                                          ANOVA 31
C           BLANK..BLANK FIELD                                                ANOVA 32
C           HEAD...FACTOR LABELS                                              ANOVA 33
C           LEVEL..LEVELS OF FACTORS                                          ANOVA 34
C           PRINT PROBLEM NUMBER AND LEVELS OF FACTORS                        ANOVA 35
      WRITE (MX,2)PR,PR1                                                      ANOVA 36
      WRITE (MX,3)(HEAD(I),LEVEL(I),I=1,K)                                    ANOVA 37
C         CALCULATE TOTAL NUMBER OF DATA ELEMENTS                             ANOVA 38
      N=LEVEL(1)                                                              ANOVA 39
      DO 102 I=2,K                                                            ANOVA 40
  102 N=N*LEVEL(I)                                                            ANOVA 41
C         READ ALL INPUT DATA                                                 ANOVA 42
      READ (MY,8)(X(I),I=1,N)                                                 ANOVA 43
      CALL AVDAT (K,LEVEL,N,X,L,ISTEP,KOUNT)                                  ANOVA 44
      CALL AVCAL (K,LEVEL,X,L,ISTEP,LASTS)                                    ANOVA 45
      CALL MEANQ (K,LEVEL,X,GMEAN,SUMSQ,NDF,SMEAN,ISTEP,KOUNT,LASTS)          ANOVA 46
C         PRINT GRAND MEAN                                                    ANOVA 47
      WRITE (MX,4)GMEAN                                                       ANOVA 48
C         PRINT ANALYSIS OF VARIANCE TABLE                                    ANOVA 49
      WRITE (MX,5)                                                            ANOVA 50
      LL=(2**K)-1                                                             ANOVA 51
      ISTEP(1)=1                                                              ANOVA 52
      DO 105 I=2,K                                                            ANOVA 53
  105 ISTEP(I)=0                                                              ANOVA 54
      DO 110 I=1,15                                                           ANOVA 55
  110 FMT(I)=BLANK                                                            ANOVA 56
      NN=0                                                                    ANOVA 57
      SUM=0.0                                                                 ANOVA 58
  120 NN=NN+1                                                                 ANOVA 59
      L=0                                                                     ANOVA 60
      DO 140 I=1,K                                                            ANOVA 61
      FMT(I)=BLANK                                                            ANOVA 62
      IF(ISTEP(I)) 130, 140, 130                                              ANOVA 63
  130 L=L+1                                                                   ANOVA 64
      FMT(L)=HEAD(I)                                                          ANOVA 65
  140 CONTINUE                                                                ANOVA 66
      WRITE (MX,6)(FMT(I),I=1,15),SUMSQ(NN),NDF(NN),SMEAN(NN)                 ANOVA 67
      SUM=SUM+SUMSQ(NN)                                                       ANOVA 68
      IF(NN-LL) 145, 170, 170                                                 ANOVA 69
  145 DO 160 I=1,K                                                            ANOVA 70
      IF(ISTEP(I)) 147, 150, 147                                              ANOVA 71
  147 ISTEP(I)=0                                                              ANOVA 72
      GO TO 160                                                               ANOVA 73
  150 ISTEP(I)=1                                                              ANOVA 74
      GO TO 120                                                               ANOVA 75
  160 CONTINUE                                                                ANOVA 76
  170 N=N-1                                                                   ANOVA 77
      WRITE (MX,7)SUM,N                                                       ANOVA 78
      GO TO 100                                                               ANOVA 79
      END                                                                     ANOVA 80
// DUP
*STORE       4S  UA  ANOVA
// XEQ ANOVA
```

```
 1 2                                                                          1
SAMPLE04        A0004B0003C0003R0002                                          2
     3   10    9   8   24   8   9   3   2   8   9   8                        3
     4   12    3   9   22   7  16   2   2   2   7   2                        4
     5   10    5   8   23   9  17   3   2   8   6   3                        5
     2   14    9  13   29  16  11   3   2   7   5   3                        6
     7   11    5   8   28  18  10   6   6   6   5   9                        7
     9   10   27   8 . 28  16  11   7   8   9   8  15                        8
```

DISCRIMINANT ANALYSIS

Problem Description

A set of linear functions is calculated from data on many groups for the purpose of classifying new individuals into one of several groups. The classification of an individual into a group is performed by evaluating each of the calculated linear functions, then finding the group for which the value is the largest.

The sample problem for discriminant analysis consists of four groups of observations as presented in Table 7. The number of observations in the first group is eight; the second group, seven; the third group, seven; and the fourth group eight. The number of variables is six in all groups.

Program

Description

The discriminant analysis sample program consists of a main routine, MDISC, and three subroutines:

DMATX
MINV
DISCR

} are from the Scientific Subroutine Package

Table 7. Sample Data for Discriminant Analysis

	Observation	X_1	X_2	X_3	X_4	X_5	X_6
Group 1	1	3	10	9	8	24	8
	2	4	12	3	8	22	7
	3	9	3	2	8	9	8
	4	16	2	2	2	7	2
	5	5	10	5	8	23	9
	6	17	3	2	8	6	3
	7	2	10	9	8	29	16
	8	7	10	5	8	28	18
Group 2	1	9	10	27	8	28	16
	2	11	7	8	9	8	15
	3	8	10	2	8	27	16
	4	1	6	8	14	14	13
	5	7	8	9	6	18	2
	6	7	9	8	2	19	9
	7	7	10	5	8	27	17
Group 3	1	3	11	9	15	20	10
	2	9	4	10	7	9	9
	3	4	13	10	7	21	15
	4	8	5	16	16	16	7
	5	6	9	10	5	23	11
	6	8	10	5	8	27	16
	7	17	3	2	7	6	3
Group 4	1	3	10	8	8	23	8
	2	4	12	3	8	23	7
	3	9	3	2	8	21	7
	4	15	2	2	2	7	2
	5	9	10	26	8	27	16
	6	8	9	2	8	26	16
	7	7	8	6	9	18	2
	8	7	10	5	8	26	16

Capacity

The capacity of the sample program and the format required for data input have been set up as follows:

1. Up to four groups
2. Up to ten variables
3. Up to a total number of 100 observations in all groups combined.
4. (12F6.0) format for input data cards

Therefore, if a problem satisfies the above conditions it is not necessary to modify the sample program. However, if there are more than four groups, more than ten variables, or more than 100 observations, dimension statements in the sample main program must be modified to handle this particular problem. Similarly, if input data cards are prepared using a different format, the input format statement in the sample main program must be modified. The general rules for program modification are described later.

Input

I/O Specification Card

One control card is required for each problem and is read by the main program, MDISC. This card is prepared as follows:

Columns	Contents	For Sample Problem
1 - 6	Problem number (may be alphameric)	SAMPLE
7 - 8	Number of groups	04
9 - 10	Number of variables	06
11 - 15	Number of observations in first group	00008
16 - 20	Number of observations in second group	00007
21 - 25	Number of observations in third group	00007
26 - 30	Number of observations in fourth group	00008
.		
.		
65 - 70	Number of observations in twelfth group (if present)	

If there are more than twelve groups in the problem, continue to the second card in the same manner.

Columns	Contents
1 - 5	Number of observations in thirteenth group
6 - 10	Number of observations in fourteenth group

Leading zeros are not required to be keypunched, but numbers must be right-justified in fields.

Data Cards

Since input data are read into the computer one observation at a time, each row of data in Table 7 is

keypunched on a separate card using the format (12F6.0). This format assumes twelve 6-column fields per card.

Deck Setup

Deck setup is shown in Figure 20.

Sample

The listing of input cards for the sample problem is presented at the end of the sample main program.

Output

Description

The output of the sample program for discriminant analysis includes:

1. Means of variables in each group
2. Pooled dispersion matrix
3. Common means
4. Generalized Mahalanobis D-square
5. Constant and coefficients of each discriminant function

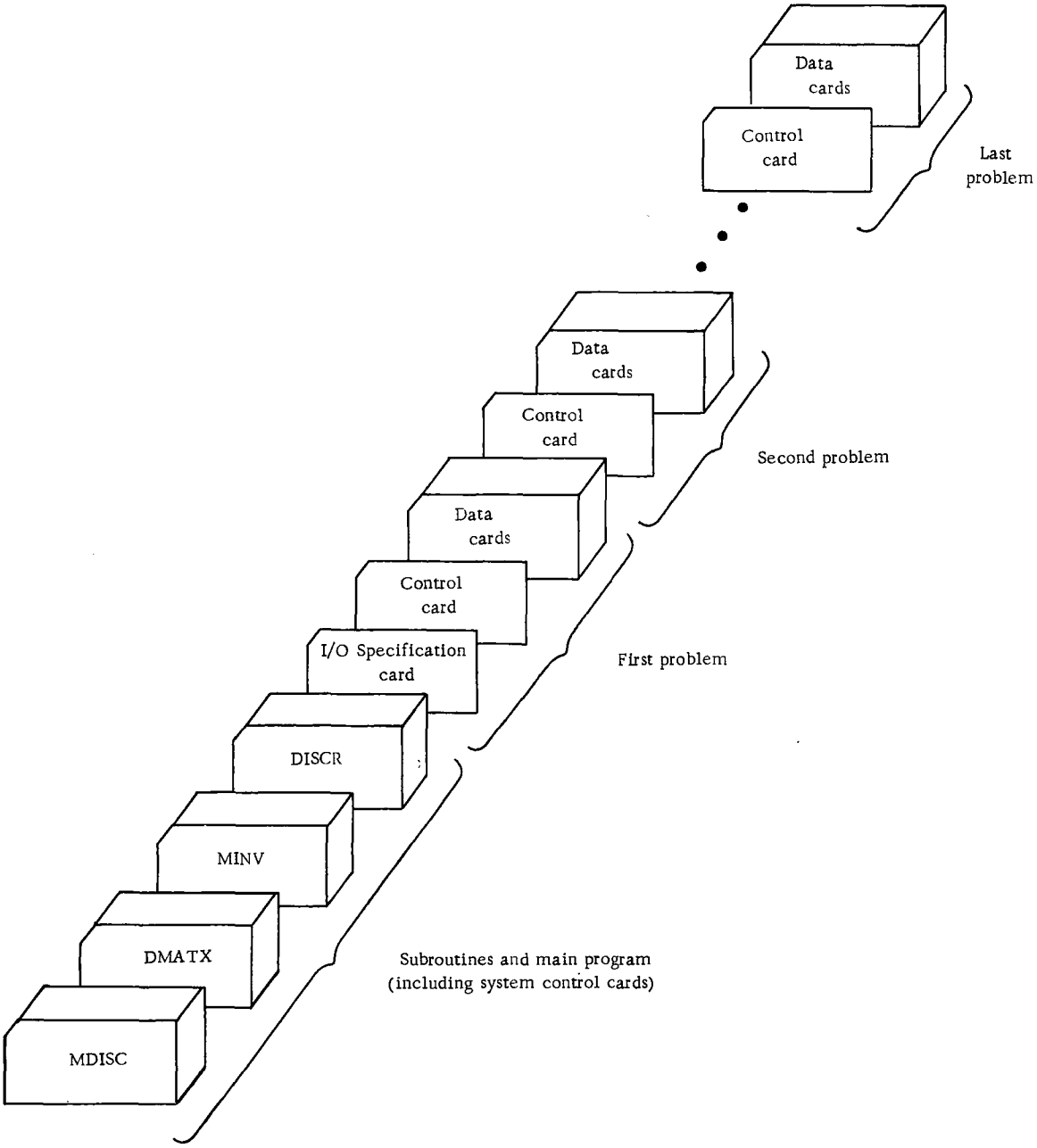


Figure 20. Deck setup (discriminant analysis)

6. Probability associated with the largest discriminant function evaluated for each observation.

Sample

The output listing for the sample problem is shown in Figure 21.

Program Modification

Noting that storage problems may result, as previously discussed in "Sample Program Description", program capacity can be increased or decreased by making changes in dimension statements. Input data in a different format can also be handled by providing a specific format statement. In order to familiarize the user with the program modification, the following general rules are supplied in terms of the sample problem:

1. Changes in the dimension statements of the main program, MDISC:
   a. The dimension of array N must be greater than or equal to the number of groups, k. Since there are four groups in the sample problem the value of k is 4.
   b. The dimension of array CMEAN must be greater than or equal to the number of variables, m. Since there are six variables in the sample problem the value of m is 6.
   c. The dimension of array XBAR must be greater than or equal to the product of m times k. For the sample problem this product is 24 = 6 x 4.
   d. The dimension of array C must be greater than or equal to the product of (m+1)k. For the sample problem this product is 28 = (6+1)4.
   e. The dimension of array D must be greater than or equal to the product of m times m. For the sample problem this product is 36 = 6 x 6.
   f. The dimension of arrays P and LG must be greater than or equal to the total number of observations in all groups combined, t. For the sample problem this total is 30 = 8 + 7 + 7 + 8.
   g. The dimension of array X must be greater than or equal to the total number of data points that is equal to the product of t

```
DISCRIMINANT ANALYSIS.....SAMPLE

  NUMBER OF GROUPS          4
  NUMBER OF VARIABLES       6
  SAMPLE SIZES..
        GROUP
          1          8
          2          7
          3          7
          4          8


GROUP  1  MEANS
      7.87500        7.50000        4.62500        7.25000       18.50000       8.87500

GROUP  2  MEANS
      7.14286        8.57143        9.57143        7.85714       20.14286      12.57143

GROUP  3  MEANS
      7.85714        7.85714        8.85714        9.28571       17.42857      10.14286

GROUP  4  MEANS
      7.75000        8.00000        6.75000        7.37500       21.37500       9.25000

POOLED DISPERSION MATRIX

ROW  1
     19.61880      -11.16208       -5.21496       -6.09889      -22.74861      -9.54051

ROW  2
    -11.16208       11.94504        5.61812        1.91758       22.60987      10.66757

ROW  3
     -5.21496        5.61812       39.45945        3.93681       16.23487       9.34546

ROW  4
     -6.09889        1.91758        3.93681        9.83309        4.62156       3.83790

ROW  5
    -22.74861       22.60987       16.23487        4.62156       62.78635      30.18268

ROW  6
     -9.54051       10.66757        9.34546        3.83790       30.18268      29.57484

COMMON MEANS
      7.66666        7.96666        7.33333        7.89999       19.39999      10.13333

GENERALIZED MAHALANOBIS D-SQUARE       12.78067

DISCRIMINANT FUNCTION  1
     CONSTANT   *   COEFFICIENTS
    -28.49425   *        2.63868        2.12202       -0.17167        1.91198       0.58476       -0.40476
DISCRIMINANT FUNCTION  2
     CONSTANT   *   COEFFICIENTS
    -29.21008   *        2.61928        2.25227       -0.04816        1.88318       0.49732       -0.21783
DISCRIMINANT FUNCTION  3
     CONSTANT   *   COEFFICIENTS
    -31.86424   *        2.74468        2.39585       -0.06456        2.13259       0.42619       -0.32718
DISCRIMINANT FUNCTION  4
     CONSTANT   *   COEFFICIENTS
    -30.82023   *        2.71858        2.03934       -0.13351        1.94538       0.71677       -0.48780

EVALUATION OF CLASSIFICATION FUNCTIONS FOR EACH OBSERVATION

GROUP  1
 OBSERVATION    PROBABILITY ASSOCIATED WITH LARGEST DISCRIMINANT FUNCTION    LARGEST FUNCTION NO.
      1                    0.38065                                              4
      2                    0.37043                                              1
      3                    0.36260                                              1
      4                    0.44189                                              1
      5                    0.34453                                              1
      6                    0.44216                                              3
      7                    0.31786                                              2
      8                    0.29272                                              2

GROUP  2
 OBSERVATION    PROBABILITY ASSOCIATED WITH LARGEST DISCRIMINANT FUNCTION    LARGEST FUNCTION NO.
      1                    0.51027                                              2
      2                    0.50061                                              3
      3                    0.34760                                              4
      4                    0.43131                                              3
      5                    0.44281                                              4
      6                    0.36406                                              2
      7                    0.28514                                              2

GROUP  3
 OBSERVATION    PROBABILITY ASSOCIATED WITH LARGEST DISCRIMINANT FUNCTION    LARGEST FUNCTION NO.
      1                    0.67612                                              3
      2                    0.46628                                              2
      3                    0.54634                                              2
      4                    0.66689                                              3
      5                    0.30599                                              2
      6                    0.33043                                              4
      7                    0.39006                                              3

GROUP  4
 OBSERVATION    PROBABILITY ASSOCIATED WITH LARGEST DISCRIMINANT FUNCTION    LARGEST FUNCTION NO.
      1                    0.33727                                              4
      2                    0.37474                                              1
      3                    0.62340                                              4
      4                    0.45696                                              1
      5                    0.52173                                              2
      6                    0.34061                                              4
      7                    0.43135                                              4
      8                    0.27848                                              1
```

Figure 21. Output listing

times m. For the sample this product is 180 = 30 x 6.

2. Changes in the input format statement of the main program, MDISC:

Only the format statement for input data may be changed. Since sample data are either one- or two-digit numbers, rather than using six-column fields as in the sample problem, each row of data may be keypunched in two-column fields, and, if so, the format is changed to (6F2.0). This format assumes six 2-column fields per card, beginning in column 1.

Operating Instructions

The sample program for discriminant analysis is a standard FORTRAN program. Special operating instructions are not required. Logical unit 2 is used for input, and logical unit 1 is used for output.

Sample Main Program for Discriminant Analysis - MDISC

Purpose:

(1) Read the problem parameter card and data for discriminant analysis, (2) Call three subroutines to calculate variable means in each group, pooled dispersion matrix, common means of variables, generalized Mahalanobis D square, coefficients of discriminant functions, and probability associated with largest discriminant function of each case in each group, and (3) Print the results.

Remarks:

The number of variables must be greater than or equal to the number of groups.
I/O logical units determined by MX and MY, respectively.

Subroutines and function subprograms required:

DMATX
MINV
DISCR

Method:

Refer to "BMD Computer Programs Manual", edited by W.J. Dixon, UCLA, 1964, and T.W. Anderson, "Introduction to Multivariate Statistical Analysis", John Wiley and Sons, 1958, section 6.6-6.8.

```
// FOR
*IOCS(CARD,TYPEWRITER,1132 PRINTER)
*ONE WORD INTEGERS
C       SAMPLE MAIN PROGRAM FOR DISCRIMINANT ANALYSIS - MDISC          MDISC  1
C       THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE   MDISC  2
C       NUMBER OF GROUPS, K..                                          MDISC  3
      DIMENSION N(4)                                                   MDISC  4
C       THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE   MDISC  5
C       NUMBER OF VARIABLES, M..                                       MDISC  6
      DIMENSION CMEAN(10)                                              MDISC  7
C       THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE   MDISC  8
C       PRODUCT OF M*K..                                               MDISC  9
      DIMENSION XBAR(40)                                               MDISC 10
C       THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE   MDISC 11
C       PRODUCT OF (M+1)*K..                                           MDISC 12
      DIMENSION C(44)                                                  MDISC 13
C       THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE   MDISC 14
C       PRODUCT OF M*M..                                               MDISC 15
      DIMENSION D(100)                                                 MDISC 16
C       THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO THE  MDISC 17
C       TOTAL OF SAMPLE SIZES OF K GROUPS COMBINED, T (T = N(1)+N(2)+..MDISC 18
C       +N(K))..                                                       MDISC 19
      DIMENSION P(100),LG(100)                                         MDISC 20
C       THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE   MDISC 21
C       TOTAL DATA POINTS WHICH IS EQUAL TO THE PRODUCT OF T*M..        MDISC 22
      DIMENSION X(1000)                                                MDISC 23
C      ..................................................................MDISC 24
    1 FORMAT(A4,A2,2I2,I2I5/(14I5))                                    MDISC 25
    2 FORMAT(////27H DISCRIMINANT ANALYSIS.....A4,A2//19H   NUMBER OF GRMDISC 26
     1OUPS,7X,I3/22H   NUMBER OF VARIABLES,I7/17H   SAMPLE SIZES../12X, MDISC 27
     25HGROUP)                                                         MDISC 28
    3 FORMAT(12X,I3,8X,I4)                                             MDISC 29
    4 FORMAT(//2X)                                                     MDISC 30
    5 FORMAT(12F6.0)                                                   MDISC 31
    6 FORMAT(//6H GROUP,I3,7H  MEANS/(8F15.5))                         MDISC 32
    7 FORMAT(///25H POOLED DISPERSION MATRIX)                          MDISC 33
    8 FORMAT(//4H ROW,I3/(8F15.5))                                     MDISC 34
    9 FORMAT(////13H COMMON MEANS/(8F15.5))                            MDISC 35
   10 FORMAT(////33H GENERALIZED MAHALANOBIS D-SQUARE,F15.5//)         MDISC 36
   11 FORMAT(//22H DISCRIMINANT FUNCTION,I3//6X,27HCONSTANT   *   COEFFIMDISC 37
     1CIENTS//F14.5,7H   *   ,7F14.5/(22X,7F14.5))                    MDISC 38
   12 FORMAT(////66H EVALUATION OF CLASSIFICATION FUNCTIONS FOR EACH OBSMDISC 39
     1ERVATION)                                                        MDISC 40
   13 FORMAT(//6H GROUP,I3/19X,27HPROBABILITY ASSOCIATED WITH,11X,7HLARGMDISC 41
     1EST/13H  OBSERVATION,5X,29HLARGEST DISCRIMINANT FUNCTION,8X,12HFUNMDISC 42
     2CTION NO.)                                                       MDISC 43
   14 FORMAT(I7,20X,F8.5,20X,I6)                                       MDISC 44
   15 FORMAT(2I2)                                                      MDISC 45
C      ..................................................................MDISC 46
      READ(2,15)MX,MY                                                  MDISC 47
C       READ PROBLEM PARAMETER CARD                                    MDISC 48
  100 READ(MY,1)PR,PR1,K,M,(N(I),I=1,K)                               MDISC 49
C       PR.......PROBLEM NUMBER (MAY BE ALPHAMERIC)                    MDISC 50
C       PR1......PROBLEM NUMBER (CONTINUED)                            MDISC 51
C       K........NUMBER OF GROUPS                                      MDISC 52
C       M........NUMBER OF VARIABLES                                   MDISC 53
C       N........VECTOR OF LENGTH K CONTAINING SAMPLE SIZES            MDISC 54
      WRITE (MX,2) PR,PR1,K,M                                          MDISC 55
      DO 110 I=1,K                                                     MDISC 56
  110 WRITE (MX,3) I,N(I)                                              MDISC 57
      WRITE (MX,4)                                                     MDISC 58
C       READ DATA                                                      MDISC 59
      L=0                                                              MDISC 60
      DO 130 I=1,K                                                     MDISC 61
      N1=N(I)                                                          MDISC 62
      DO 120 J=1,N1                                                    MDISC 63
      READ (MY,5) (CMEAN(IJ),IJ=1,M)                                   MDISC 64
      L=L+1                                                            MDISC 65
      N2=L-N1                                                          MDISC 66
      DO 120 IJ=1,M                                                    MDISC 67
      N2=N2+N1                                                         MDISC 68
  120 X(N2)=CMEAN(IJ)                                                  MDISC 69
  130 L=N2                                                             MDISC 70
      CALL DMATX (K,M,N,X,XBAR,D,CMEAN)                                MDISC 71
C       PRINT MEANS AND POOLED DISPERSION MATRIX                       MDISC 72
      L=0                                                              MDISC 73
      DO 150 I=1,K                                                     MDISC 74
      DO 140 J=1,M                                                     MDISC 75
      L=L+1                                                            MDISC 76
  140 CMEAN(J)=XBAR(L)                                                 MDISC 77
  150 WRITE (MX,6) I,(CMEAN(J),J=1,M)                                  MDISC 78
      WRITE (MX,7)                                                     MDISC 79
      DO 170 I=1,M                                                     MDISC 80
      L=I-M                                                            MDISC 81
      DO 160 J=1,M                                                     MDISC 82
      L=L+M                                                            MDISC 83
  160 CMEAN(J)=D(L)                                                    MDISC 84
  170 WRITE (MX,8) I,(CMEAN(J),J=1,M)                                  MDISC 85
      CALL MINV (D,M,DET,CMEAN,C)                                      MDISC 86
      CALL DISCR (K,M,N,X,XBAR,D,CMEAN,V,C,P,LG)                       MDISC 87
C       PRINT COMMON MEANS                                             MDISC 88
      WRITE(MX,9) (CMEAN(I),I=1,M)                                     MDISC 89
C       PRINT GENERALIZED MAHALANOBIS D-SQUARE                         MDISC 90
      WRITE (MX,10) V                                                  MDISC 91
C       PRINT CONSTANTS AND COEFFICIENTS OF DISCRIMINANT FUNCTIONS     MDISC 92
      N1=1                                                             MDISC 93
      N2=M+1                                                           MDISC 94
      DO 180 I=1,K                                                     MDISC 95
      WRITE (MX,11) I,(C(J),J=N1,N2)                                   MDISC 96
      N1=N1+(M+1)                                                      MDISC 97
  180 N2=N2+(M+1)                                                      MDISC 98
C       PRINT EVALUATION OF CLASSIFICATION FUNCTIONS FO EACH           MDISC 99
C       OBSERVATION                                                    MDISC100
      WRITE (MX,12)                                                    MDISC101
      N1=1                                                             MDISC102
      N2=N(1)                                                          MDISC103
      DO 210 I=1,K                                                     MDISC104
      WRITE (MX,13) I                                                  MDISC105
      L=0                                                              MDISC106
      DO 190 J=N1,N2                                                   MDISC107
      L=L+1                                                            MDISC108
  190 WRITE (MX,14) L,P(J),LG(J)                                       MDISC109
      IF(I-K) 200, 100, 100                                            MDISC110
  200 N1=N1+N(I)                                                       MDISC111
      N2=N2+N(I+1)                                                     MDISC112
  210 CONTINUE                                                         MDISC113
      STOP                                                             MDISC114
      END                                                              MDISC115
// DUP
*STORE      WS  UA  MDISC
// XEQ MDISC   01
*LOCALMDISC,DMATX,MINV,DISCR
```

```
 1 2
SAMPLE040600008000070000700008                                         1
     3    10     9     8    24     8                                   2
     4    12     3     8    22     7                                   3
     9     3     2     8     9     8                                   4
    16     2     2     2     7     2                                   5
     5    10     5     8    23     9                                   6
    17     3     2     8     6     3                                   7
     2    10     9     8    29    16                                   8
     7    10     5     8    28    18                                   9
     9    10    27     8    28    16                                   10
    11     7     8     9     8    15                                   11
     8    10     2     8    27    16                                   12
     1     6     8    14    14    13                                   13
     7     8     9     6    18     2                                   14
                                                                       15
```

```
 7   9   8   2  19   9
 7  10   5   8  27  17
 3  11   9  15  20  10
 9   4  10   7   9   9
 4  13  10   7  21  15
 8   5  16  16  16   7
 6   9  10   5  23  11
 8  10   5   8  27  16
17   3   2   7   6   3
 3  10   8   8  23   8
 4  12   3   8  23   7
 9   3   2   8  21   7
15   2   2   2   7   2
 9  10  26   8  27  16
 8   9   2   8  26  16
 7   8   6   9  18   2
 7  10   5   8  26  16
```

FACTOR ANALYSIS

Problem Description

A principal component solution and the varimax rotation of the factor matrix are performed. Principal component analysis is used to determine the minimum number of independent dimensions needed to account for most of the variance in the original set of variables. The varimax rotation is used to simplify columns (factors) rather than rows (variables) of the factor matrix.

The sample problem for factor analysis consists of 23 observations with nine variables as presented in Table 8. In order to keep the number of independent dimensions as small as possible, only those eigenvalues (of correlation coefficients) greater than or equal to 1.0 are retained in the analysis.

Table 8. Sample Data for Factor Analysis

Observation	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
1	7	7	9	7	15	36	60	15	24
2	13	18	25	15	13	35	61	18	30
3	9	18	24	23	12	43	62	14	31
4	7	13	25	36	11	12	63	26	32
5	6	8	20	7	15	46	18	28	15
6	10	12	30	11	10	42	27	12	17
7	7	6	11	7	15	35	60	20	25
8	16	19	25	16	13	30	64	20	30
9	9	22	26	24	13	40	66	15	32
10	8	15	26	30	13	10	66	25	34
11	8	10	20	8	17	40	20	30	18
12	9	12	28	11	8	45	30	15	19
13	11	17	21	30	10	45	60	17	30
14	9	16	26	27	14	31	59	19	17
15	10	15	24	18	12	29	48	18	26
16	11	11	30	19	19	26	57	20	30
17	16	9	16	20	18	31	60	21	17
18	9	8	19	14	16	33	67	9	19
19	7	18	22	9	15	37	62	11	20
20	8	11	23	18	9	36	61	22	24
21	6	6	27	23	7	40	55	24	31
22	10	9	26	26	10	37	57	27	29
23	8	10	26	15	11	42	59	20	28

Program

Description

The factor analysis sample program consists of a main routine, FACTO, and six subroutines:

CORRE	
EIGEN	are from the Scientific Subroutine
TRACE	Package
LOAD	
VARMX	
DATA	is a special input subroutine

Capacity

The capacity of the sample program and the format required for data input have been set up as follows:

1. Up to 29 variables
2. Up to 99,999 observations
3. (12F6.0) format for input data cards

Therefore, if a problem satisfies the above conditions it is not necessary to modify the sample program. However, if there are more than 30 variables, dimension statements in the sample main program must be modified to handle this particular problem. Similarly, if input data cards are prepared using a different format, the input format statement in the input subroutine, DATA, must be modified. The general rules for program modification are described later.

Input

I/O Specification Card

One control card is required for each problem and is read by the main program, FACTO. This card is prepared as follows:

Columns	Contents	For Sample Problem
1 - 6	Problem number (may be alphameric)	SAMPLE
7 - 11	Number of observations	00023
12 - 13	Number of variables	09
14 - 19	Value used to limit the number of eigenvalues of	0001.0

Columns	Contents	For Sample Problem
14 - 19 (cont)	correlation coefficients. Only those eigenvalues greater than or equal to this value are retained in the analysis. (A decimal point must be specified.)	

Leading zeros are not required to be keypunched, but numbers must be right-justified in fields.

Data Cards

Since input data are read into the computer one observation at a time, each row of data in Table 8 is keypunched on a separate card using the format (12F6.0). This format assumes twelve 6-column fields per card.

If there are more than twelve variables in a problem, each row of data is continued on the second and third cards until the last data point is keypunched. However, each row of data must begin on a new card.

Deck Setup

Deck setup is shown in Figure 22.

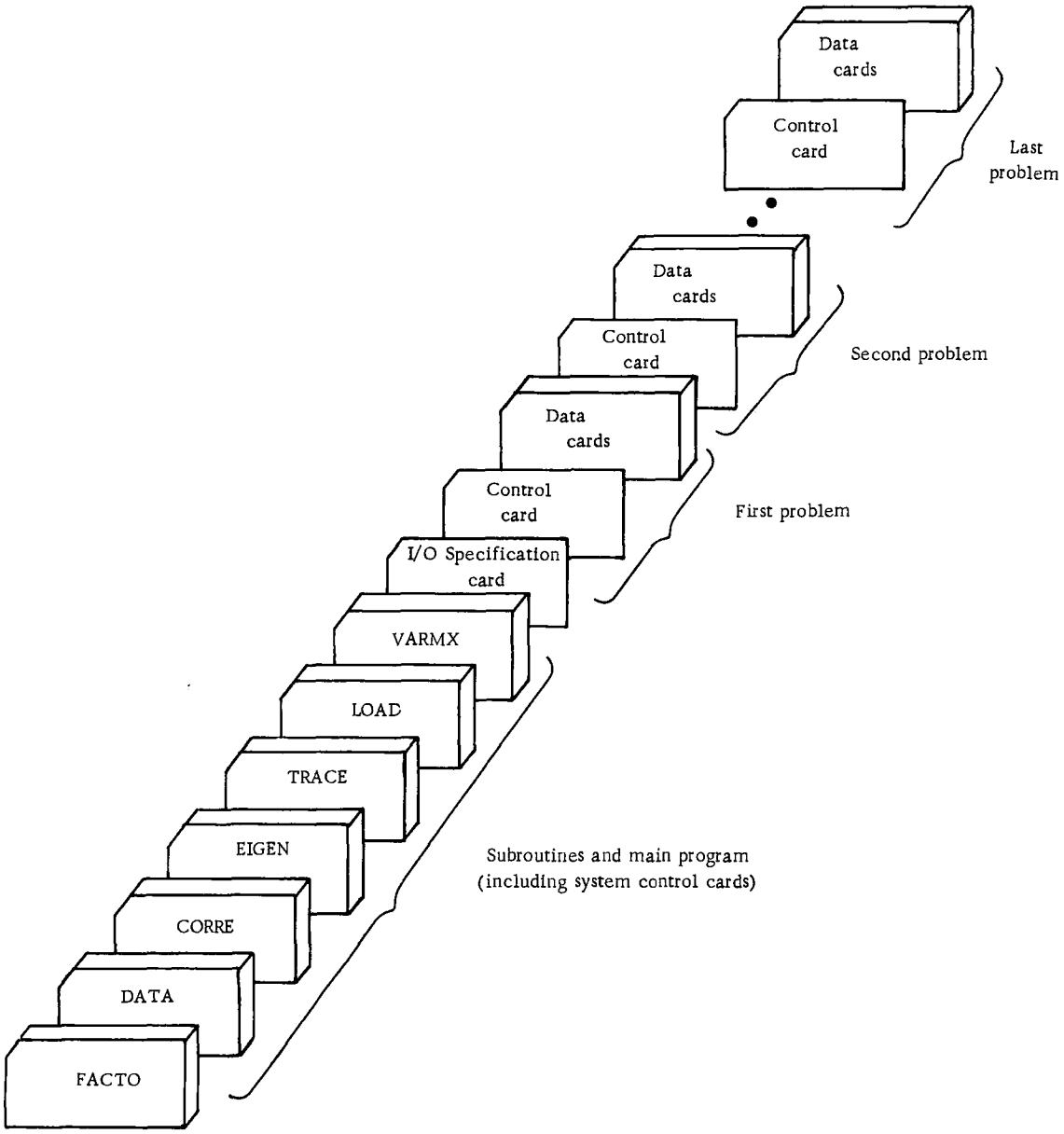


Figure 22. Deck setup (factor analysis)

Sample

The listing of input cards for the sample problem is presented at the end of the sample main program.

Output

Description

The output of the sample program for factor analysis includes:

1. Means
2. Standard deviations
3. Correlation coefficients
4. Eigenvalues
5. Cumulative percentage of eigenvalues
6. Eigenvectors
7. Factor matrix
8. Variance of the factor matrix for each iteration cycle
9. Rotated factor matrix
10. Check on communalities

Sample

The output listing for the sample problem is shown in Figure 23.

Program Modification

Noting that storage problems may result, as previously described in "Sample Program Description", program capacity can be increased or decreased by making changes in dimension statements. Input data in a different format can also be handled by providing a specific format statement. In order to familiarize the user with the program modification, the following general rules are supplied in terms of the sample problem:

1. Changes in the dimension statements of the main program, FACTO:
   a. The dimension of arrays B, D, S, T, and XBAR must be greater than or equal to the number of variables, m. Since there are nine variables in the sample problem the value of m is 9.
   b. The dimension of array V must be greater than or equal to the product of m times m. For the sample problem this product is 81 = 9 x 9.
   c. The dimension of array R must be greater than $\frac{(m+1)m}{2}$

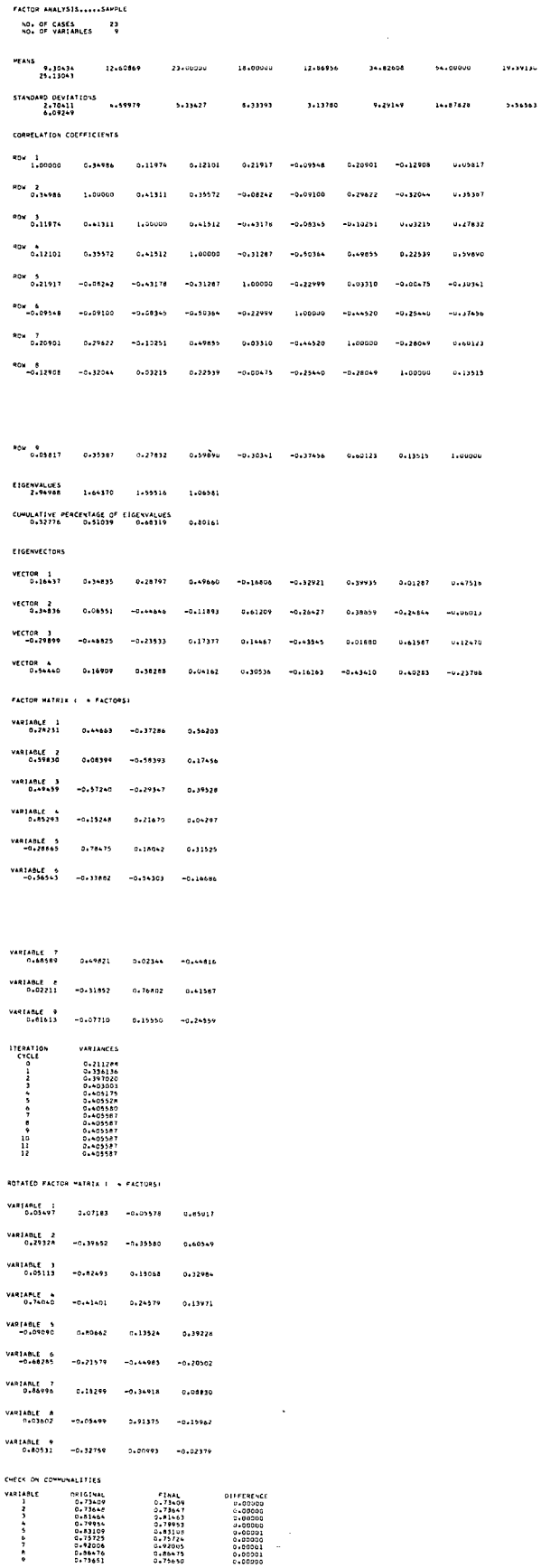


Figure 23. Output listing

For the sample problem, this number is

$$45 = \frac{(9+1)9}{2}$$

2. Changes in the input format statement of the special input subroutine, DATA:

a. Only the format statement for input data may be changed. Since sample data are either one- or two-digit numbers, rather than using six-column fields as in the sample problem, each row of data may be keypunched in two-column fields, and, if so, the format is changed to (9F2.0). This format assumes nine 2-column fields per card, beginning in column 1.

b. The special input subroutine, DATA, is normally written by the user to handle different formats for different problems. The user may modify this subroutine to perform testing of input data, transformation of data, and so on.

Operating Instructions

The sample program for factor analysis is a standard FORTRAN program. Special operating instructions are not required. Logical unit 2 is used for input, and logical unit 1 is used for output.

Error Messages

If the number of factors to be rotated is one or zero, the following message will be printed:

ONLY____FACTOR RETAINED, NO ROTATION.

The program skips rotation and goes to the next problem if it is present.

Sample Main Program for Factor Analysis - FACTO

Purpose:

(1) Read the problem parameter card, (2) Call five subroutines to perform a principal component solution and the varimax rotation of a factor matrix, and (3) Print the results.

Remarks:

I/O specifications transmitted to subroutines by COMMON.

Input card:

Column 2 MX - Logical unit number for output.

Column 4 MY - Logical unit number for input.

Subroutines and function subprograms required:

(which, in turn, calls the subroutine named DATA.)

CORRE
EIGEN
TRACE
LOAD
VARMX

Method:

Refer to "BMD Computer Programs Manual", edited by W. J. Dixon, UCLA, 1964.

```
// FOR
*IOCS(CARD,TYPEWRITER,1132 PRINTER)
*ONE WORD INTEGERS
C        SAMPLE MAIN PROGRAM FOR FACTOR ANALYSIS - FACTO                  FACTO  1
C        THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO THE    FACTO  2
C        NUMBER OF VARIABLES, M..                                         FACTO  3
      DIMENSION B(29),D(29),S(29),T(29),XBAR(29)                          FACTOMO1
C        THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE     FACTO  5
C        PRODUCT OF M*M..                                                 FACTO  6
      DIMENSION V(841)                                                    FACTOMO2
C        THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO         FACTO  8
C        (M+1)*M/2..                                                      FACTO  9
      DIMENSION R(435)                                                    FACTOMO3
C        THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO 51..    FACTO 11
         DIMENSION TV(51)                                                 FACTO 12
      COMMON MX,MY                                                        FACTO 13
    1 FORMAT(////21H FACTOR ANALYSIS.....A4,A2//3X,12HNO. OF CASES,4X,    FACTO 14
     1I6/3X,16HNO. OF VARIABLES,I6/)                                      FACTO 15
    2 FORMAT(//6H MEANS/(8F15.5))                                         FACTO 16
    3 FORMAT(//20H STANDARD DEVIATIONS/(8F15.5))                          FACTO 17
    4 FORMAT(//25H CORRELATION COEFFICIENTS)                              FACTO 18
    5 FORMAT(//4H ROW,I3/(10F12.5))                                       FACTO 19
    6 FORMAT(///12H EIGENVALUES/(10F12.5))                                FACTO 20
    7 FORMAT(//37H CUMULATIVE PERCENTAGE OF EIGENVALUES/(10F12.5))        FACTO 21
    8 FORMAT(///13H EIGENVECTORS)                                         FACTO 22
    9 FORMAT(//7H VECTOR,I4/(10F12.5))                                    FACTO 23
   10 FORMAT(///16H FACTOR MATRIX (,I3,9H FACTORS))                       FACTO 24
   11 FORMAT(//9H VARIABLE,I3/(10F12.5))                                  FACTO 25
   12 FORMAT(///10H ITERATION,7X, 9HVARIANCE,8H   CYCLE)                  FACTO 26
   13 FORMAT(I6,F20.6)                                                    FACTO 27
   14 FORMAT(///24H ROTATED FACTOR MATRIX (,I3,9H FACTORS))               FACTO 28
   15 FORMAT(//9H VARIABLE,I3/(10F12.5))                                  FACTO 29
   16 FORMAT(///23H CHECK ON COMMUNALITIES//9H VARIABLE,7X,8HORIGINAL,    FACTO 30
     112X,5HFINAL,10X,10HDIFFERENCE)                                      FACTO 31
   17 FORMAT(I6,3F18.5)                                                   FACTO 32
   18 FORMAT(A4,A2,I5,I2,F6.0)                                            FACTO 33
   19 FORMAT(//5H ONLY,I2,30H FACTOR RETAINED. NO ROTATION )              FACTO 34
   20 FORMAT(2I2)                                                         FACTO 35
      READ(2,20)MX,MY                                                     FACTO 36
C        READ PROBLEM PARAMETER CARD                                      FACTO 37
  100 READ (MY,18)PR,PR1,N,M,CON                                         FACTO 38
C        PR.........PROBLEM NUMBER (MAY BE ALPHAMERIC)                    FACTO 39
C        PR1........PROBLEM NUMBER (CONTINUED)                            FACTO 40
C        N..........NUMBER OF CASES                                       FACTO 41
C        M..........NUMBER OF VARIABLES                                   FACTO 42
C        CON........CONSTANT USED TO DECIDE HOW MANY EIGENVALUES          FACTO 43
C                   TO RETAIN                                            FACTO 44
      WRITE (MX,1)PR,PR1,N,M                                              FACTO 45
      IO=0                                                                FACTO 46
      X=0.0                                                               FACTO 47
      CALL CORRE (N,M,IO,X,XBAR,S,V,R,D,d,T)                              FACTO 48
C        PRINT MEANS                                                      FACTO 49
      WRITE (MX,2)(XBAR(J),J=1,M)                                         FACTO 50
C        PRINT STANDARD DEVIATIONS                                        FACTO 51
      WRITE (MX,3)(S(J),J=1,M)                                            FACTO 52
C        PRINT CORRELATION COEFFICIENTS                                   FACTO 53
      WRITE (MX,4)                                                        FACTO 54
      DO 120 I=1,M                                                        FACTO 55
      DO 110 J=1,M                                                        FACTO 56
      IF(I-J) 102, 104, 104                                               FACTO 57
  102 L=I+(J*J-J)/2                                                       FACTO 58
      GO TO 110                                                           FACTO 59
  104 L=J+(I*I-I)/2                                                       FACTO 60
  110 D(J)=R(L)                                                           FACTO 61
  120 WRITE (MX,5)I,(D(J),J=1,M)                                          FACTO 62
      MV=0                                                                FACTO 63
      CALL EIGEN (R,V,M,MV)                                               FACTO 64
      CALL TRACE (M,R,CON,K,D)                                            FACTO 65
C        PRINT EIGENVALUES                                                FACTO 66
      DO 130 I=1,K                                                        FACTO 67
      L=I+(I*I-I)/2                                                       FACTO 68
  130 S(I)=R(L)                                                           FACTO 69
      WRITE (MX,6)(S(J),J=1,K)                                            FACTO 70
C        PRINT CUMULATIVE PERCENTAGE OF EIGENVALUES                       FACTO 71
      WRITE (MX,7)(D(J),J=1,K)                                            FACTO 72
C        PRINT EIGENVECTORS                                               FACTO 73
      WRITE (MX,8)                                                        FACTO 74
      L=0                                                                 FACTO 75
      DO 150 J=1,K                                                        FACTO 76
      DO 140 I=1,M                                                        FACTO 77
      L=L+1                                                               FACTO 78
  140 D(I)=V(L)                                                           FACTO 79
  150 WRITE (MX,9)J,(D(I),I=1,M)                                          FACTO 80
      CALL LOAD (M,K,R,V)                                                 FACTO 81
C        PRINT FACTOR MATRIX                                              FACTO 82
      WRITE (MX,10)K                                                      FACTO 83
      DO 180 I=1,M                                                        FACTO 84
      DO 170 J=1,K                                                        FACTO 85
      L=M*(J-1)+I                                                         FACTO 86
  170 D(J)=V(L)                                                           FACTO 87
  180 WRITE (MX,11)I,(D(J),J=1,K)                                         FACTO 88
      IF(K-1) 185, 185, 188                                               FACTO 89
  185 WRITE (MX,19)K                                                      FACTO 90
      GO TO 100                                                           FACTO 91
  188 CALL VARMX (M,K,V,NC,TV,B,T,D)                                      FACTO 92
```

```
C        PRINT VARIANCES                                                FACTO 93
         NV=NC+1                                                        FACTO 94
         WRITE (MX,12)                                                  FACTO 95
         DO 190 I=1,NV                                                  FACTO 96
         NC=I-1                                                         FACTO 97
  190 WRITE (MX,13)NC,TV(I)                                             FACTO 98
C        PRINT ROTATED FACTOR MATRIX                                    FACTO 99
         WRITE (MX,14)K                                                 FACTO100
         DO 220 I=1,M                                                   FACTO101
         DO 210 J=1,K                                                   FACTO102
         L=M*(J-1)+I                                                    FACTO103
  210 S(J)=V(L)                                                         FACTO104
  220 WRITE(MX,15)I,(S(J),J=1,K)                                        FACTO105
C        PRINT COMMUNALITIES                                            FACTO106
         WRITE (MX,16)                                                  FACTO107
         DO 230 I=1,M                                                   FACTO108
  230 WRITE (MX,17)I,B(I),T(I),D(I)                                     FACTO109
         GO TO 100                                                      FACTO110
         END                                                            FACTO111
// DUP
*STORE      WS  UA  FACTO
// XEQ FACTO   01
*LOCALFACTO,CORRE,EIGEN,TRACE,LOAD,VARMX
```

```
 1 2                                                                    1
SAMPLE0002309000l.0                                                     2
     7    7    9    7   15   36   60   15   24                         3
    13   18   29   15   13   35   61   18   30                         4
     9   18   24   23   12   43   62   14   31                         5
     7   13   25   36   11   12   63   26   32                         6
     6    8   20    7   15   46   18   28   15                         7
    10   12   30   11   10   42   27   12   17                         8
     7    6   11    7   15   35   60   20   25                         9
    16   19   25   16   13   30   64   20   30                        10
     9   22   26   24   13   40   66   15   32                        11
     8   15   26   30   13   10   66   25   34                        12
     8   10   20    8   17   40   20   30   18                        13
     9   12   28   11    8   43   30   15   19                        14
    11   17   21   30   10   45   60   17   30                        15
     9   16   26   27   14   31   59   19   17                        16
    10   15   24   18   12   29   48   18   26                        17
    11   11   30   19   19   26   57   20   30                        18
    16    9   16   20   18   31   60   21   17                        19
     9    8   19   14   16   33   67    9   19                        20
     7   18   22    9   15   37   62   11   20                        21
     8   11   23   18    9   36   61   22   24                        22
     6    6   27   23    7   40   55   24   31                        23
    10    9   26   26   10   37   57   27   29                        24
     8   10   26   15   11   42   59   20   28                        25
```

```
      SAMPLE INPUT SUBROUTINE - DATA

      PURPOSE
         READ AN OBSERVATION (N DATA VALUES) FROM INPUT DEVICE.
         THIS SUBROUTINE IS CALLED BY THE SUBROUTINE CORRE AND MUST
         BE PROVIDED BY THE USER.  IF SIZE AND LOCATION OF DATA
         FIELDS ARE DIFFERENT FROM PROBLEM TO PROBLEM, THIS SUB-
         ROUTINE MUST BE RECOMPILED WITH A PROPER FORMAT STATEMENT.

      USAGE
         CALL DATA (N,D)

      DESCRIPTION OF PARAMETERS
         N - THE NUMBER OF VARIABLES IN AN OBSERVATION.
         D - OUTPUT VECTOR OF LENGTH N CONTAINING THE OBSERVATION
             DATA.

      REMARKS
         THE TYPE OF CONVERSION SPECIFIED IN THE FORMAT MUST BE
         EITHER F OR E.

      SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
         NONE
```

```
      SUBROUTINE DATA (N,D)                                             DATA  1
      DIMENSION D(1)                                                    DATA  2
      COMMON MX,MY                                                      DATA  3
    1 FORMAT(12F6.0)                                                    DATA  4
C        READ AN OBSERVATION FROM INPUT DEVICE.                         DATA  5
      READ (MY,1) (D(I),I=1,N)                                          DATA  6
      RETURN                                                            DATA  7
      END                                                               DATA  8
```

TRIPLE EXPONENTIAL SMOOTHING

Problem Description

Given a time series X, a smoothing constant, and three coefficients of the prediction equation, this sample program finds the triple exponentially smoothed series S of the time series X.

Program

Description

The sample program for triple exponential smoothing consists of a main routine, EXPON, and one subroutine, EXSMO, from the Scientific Subroutine Package.

Capacity

The capacity of the sample program and the format required for data input have been set up as follows:

1. Up to 1000 data points in a given time series
2. (12F6.0) format for input data cards

Therefore, if a problem satisfies the above conditions it is not necessary to modify the sample program. However, if there are more than 1000 data points, the dimension statement in the sample main program must be modified to handle this particular problem. Similarly, if input data cards are prepared using a different format, the input format in the sample main program must be modified. The general rules for program modification are described later.

Input

I/O Specification Card

One control card is required for each problem and is read by the main program, EXPON. This card is prepared as follows:

Columns	Contents	For Sample Problem
1 - 6	Problem number (may be alphameric)	SAMPLE
7 - 10	Number of data points in a given time series	0038
11 - 15	Smoothing constant, $(0.0 < \alpha < 1.0)$	0.1
16 - 25	First coefficient (A) of the prediction equation	0.0
26 - 35	Second coefficient (B) of the prediction equation	0.0
36-45	Third coefficient (C) of the prediction equation	0.0

Leading zeros are not required to be keypunched, but numbers must be right-justified in fields.

Data Cards

Time series data are keypunched using the format (12F6.0). This format assumes that each data point is keypunched in a six-column field and twelve fields per card.

Deck Setup

Deck setup is shown in Figure 24.

Sample

The listing of input cards for the sample problem is presented at the end of the sample main program.

Output

Description

The output of the sample program for triple exponential smoothing includes:

1. Original and updated coefficients
2. Time series as input and triple exponentially smoothed time series.

Sample

The output listing for the sample problem is shown in Figure 25.

Program Modification

Noting that storage problems may result, as previously discussed in "Sample Program Description", program capacity can be increased or decreased by making changes in the dimension statement. Input data in a different format can also be handled by providing a specific format statement. In order to familiarize the user with the program modification, the following general rules are supplied in terms of the sample problem:

1. Changes in the dimension statement of the main program, EXPON:

The dimension of arrays X and S must be greater than or equal to the number of data points in time series, NX. Since there are 38 data points in the sample problem, the value of NX is 38.

2. Changes in the input format statement of the main program, EXPON:

Only the format statement for input data may be changed. Since sample data are three-digit numbers, rather than using six-column fields as in the sample program, each data point may be keypunched in a three-column field and 24 fields per card. If so, the format is changed to (24F3.0).

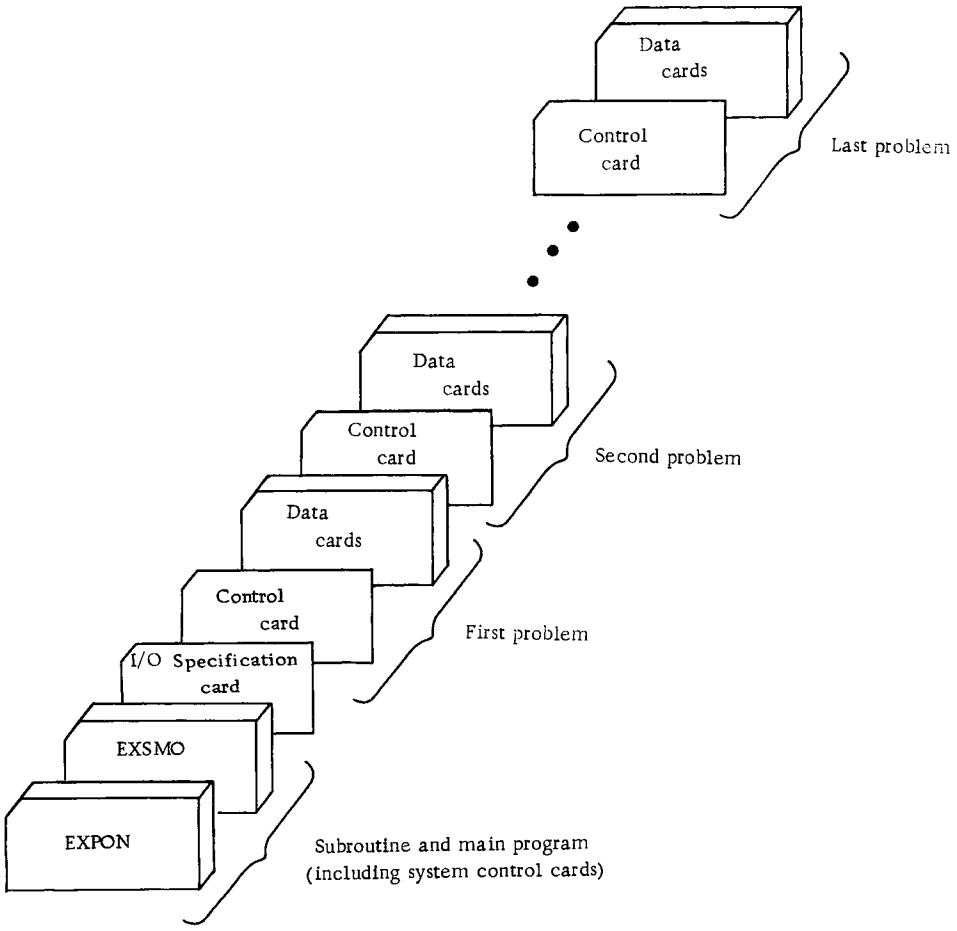


Figure 24. Deck setup (triple exponential smoothing)

```
TRIPLE EXPONENTIAL SMOOTHING.....SAMPLE

NUMBER OF DATA POINTS    38
SMOOTHING CONSTANT     0.100


COEFFICIENTS          A              B              C

ORIGINAL          0.00000        0.00000        0.00000

UPDATED         484.80169        1.71278        0.04165


                              SMOOTHED DATA
      INPUT DATA               (FORECAST)
       430.00006                430.00006
       426.00006                426.00006
       422.00006                422.00006
       419.00006                418.00006
       414.00006                414.29998
       413.00006                410.23993
       412.00006                407.08990
       409.00006                404.66839
       411.00006                402.22406
       417.00006                401.25134
       422.00006                402.64642
       430.00006                405.61694
       438.00006                410.71417
       441.00006                417.47027
       447.00006                423.99908
       455.00006                431.18335
       461.00006                439.43420
       453.00006                447.87902
       448.00006                452.21600
       449.00006                454.10571
       454.00006                455.80731
       463.00006                458.54632
       470.00006                463.30535
       472.00006                469.06439
       476.00006                474.09521
       481.00006                479.11016
       483.00006                484.38598
       487.00006                488.94592
       491.00006                493.50836
       492.00006                498.05432
       485.00006                501.66992
       486.00006                502.12536
       482.00006                502.44427
       479.00006                501.16723

       479.00006                498.92730
       476.00006                496.84124
       472.00006                494.00787
       470.00006                490.30413
```

Figure 25. Output listing

Operating Instructions

The sample program for triple exponential smoothing is a standard FORTRAN program. Special operating instructions are not required. Logical unit 2 is used for input, and logical unit 1 is used for output.

Sample Main Program for Triple Exponential Smoothing - EXPON

Purpose:

(1) Read the problem parameter card and a time series, (2) Call the subroutine EXSMO to smooth the time series, and (3) Print the result.

Remarks:

A smoothing constant specified in the problem parameter card must be greater than zero but less than one in order to obtain reasonable results.

I/O logical units determined by MX and MY, respectively.

Subroutines and function subprograms required:

EXSMO.

Method:

Refer to R. G. Brown, "Smoothing, Forecasting and Prediction of Discrete Time Series", Prentice-Hall, N. J., 1963, pp. 140 to 144.

```
// FOR
*IOCS(CARD,TYPEWRITER,1132 PRINTER)
*ONE WORD INTEGERS
C         SAMPLE MAIN PROGRAM FOR TRIPLE EXPONENTIAL SMOOTHING - EXPON    EXPON  1
C         THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE    EXPON  2
C         NUMBER OF DATA POINTS IN A GIVEN TIME SERIES.                   EXPON  3
      DIMENSION X(1000),S(1000)                                           EXPON  4
    1 FORMAT(A4,A2,I4,F5.0,3F10.0)                                        EXPON  5
    2 FORMAT(12F6.0)                                                      EXPON  6
    3 FORMAT(////34H TRIPLE EXPONENTIAL SMOOTHING.....,A4,A2//22H NUMBER EXPON  7
     1 OF DATA POINTS,I6/19H SMOOTHING CONSTANT,F9.3/)                    EXPON  8
    4 FORMAT(//13H COEFFICIENTS,9X,1HA,14X,1HB,14X,1HC)                   EXPON  9
    5 FORMAT(//9H ORIGINAL,F19.5,2F15.5)                                  EXPON 10
    6 FORMAT(//8H UPDATED,F20.5,2F15 .5/)                                 EXPON 11
    7 FORMAT(//27X,13HSMOOTHED DATA/7X,10HINPUT DATA,12X,10H(FORECAST))   EXPON 12
    8 FORMAT(F17.5,8X,F15.5)                                              EXPON 13
    9 FORMAT(2I2)                                                         EXPON 14
      READ(2,9)MX,MY                                                      EXPON 15
C         READ PROBLEM PARAMETER CARD                                     EXPON 16
  100 READ (MY,1) PK,PR1,NX,AL,A,B,C                                      EXPON 17
C         PR......PROBLEM NUMBER (MAY BE ALPHAMERIC)                      EXPON 18
C         PK1.....PROBLEM NUMBER (CONTINUED)                              EXPON 19
C         NX......NUMBER OF DATA POINTS IN TIME SERIES                    EXPON 20
C         AL......SMOOTHING CONSTANT                                      EXPON 21
C         A,B,C...COEFFICIENTS OF THE PREDICTION EQUATION                 EXPON 22
      WRITE (MX,3) PR,PR1,NX,AL                                           EXPON 23
C         PRINT ORIGINAL COEFFICIENTS                                     EXPON 24
      WRITE (MX,4)                                                        EXPON 25
      WRITE (MX,5) A,B,C                                                  EXPON 26
C         READ TIME SERIES DATA                                           EXPON 27
      READ (MY,2) (X(I),I=1,NX)                                           EXPON 28
      CALL EXSMO (X,NX,AL,A,B,C,S)                                        EXPON 29
C         PRINT UPDATED COEFFICIENTS                                      EXPON 30
      WRITE (MX,6) A,B,C                                                  EXPON 31
C         PRINT INPUT AND SMOOTHED DATA                                   EXPON 32
      WRITE (MX,7)                                                        EXPON 33
      DO 200 I=1,NX                                                       EXPON 34
  200 WRITE (MX,8) X(I),S(I)                                              EXPON 35
      GO TO 100                                                           EXPON 36
      END                                                                 EXPON 37
// DUP
*STORE      WS  UA  EXPON
// XEQ EXPON
```

```
1 2
SAMPLE  38  0.1      0.0      0.0      0.0                                  1
  430   426   422   419   414   413   412   409   411   417   422   430    2
  438   441   447   455   461   453   448   449   454   463   470   472    3
  476   481   483   487   491   492   485   486   482   479   479   476    4
  472   470                                                               5
                                                                          6
```

MATRIX ADDITION

Problem Description

An input matrix is added to another input matrix to form a resultant matrix. Each set of input matrices

and the corresponding output matrix is printed. The procedure is repeated until all sets of input matrices have been processed.

Program

Description

The matrix addition sample program consists of a main routine, ADSAM, and four subroutines:

MADD, LOC	are from the Scientific Subroutine Package
MATIN, MXOUT	are sample subroutines for matrix input and output

Capacity

Matrix size has arbitrarily been set at 650 data elements. Therefore, if a problem satisfies the above condition, no modification in the sample program is necessary. However, if there are more than 650 elements, the dimension statement in the sample main program must be modified to handle this particular problem. The general rules for program modification are described later.

Input

I/O Specification Card

Each input matrix must be preceded by a control card with the following format:

Columns	Contents	For Sample Problem
1 - 2	Blank	
3 - 6	Up to four-digit identification code	0001
7 - 10	Number of rows in matrix	0008
11 - 14	Number of columns in matrix	0011
15 - 16	Storage mode of matrix 0 for general matrix 1 for symmetric matrix 2 for diagonal matrix	0

Each input matrix must be followed by a card with a 9-punch in column 1.

Data Cards

Data cards are assumed to have seven fields of ten columns each. The decimal point may appear anywhere in a field, or may be omitted; however, all numbers must be right-justified. The number in each field may be preceded by blanks. Data elements must be punched by row. A row may continue from card to card. However, each new row must start in the first field of the next card. Only the upper triangular portion of a symmetric or the diagonal elements of a diagonal matrix are contained on data cards. The first element of each new row will be the diagonal element for a matrix with symmetric or diagonal storage mode. Columns 71-80 of data cards may be used for identification, sequence numbering, etc.

A blank card after the last pair of input matrices terminates the run.

Deck Setup

The deck setup is shown in Figure 26.

Sample

A listing of input cards for the sample problem is presented at the end of the sample main program.

Output

Description

Both sets of input matrices and the output matrix are printed. The resultant matrix is printed for any sized array as a general matrix regardless of the storage mode. Each seven-column grouping is headed with the matrix code number, dimensions, and storage mode. Columns and rows are headed with their respective number. The code number for the output matrix is derived by adding the code numbers for the input matrices.

Sample

The output listing for the sample problem is shown in Figure 27.

Program Modification

Noting that storage problems may result, as previously described in "Sample Program Description", the maximum matrix size acceptable to the sample

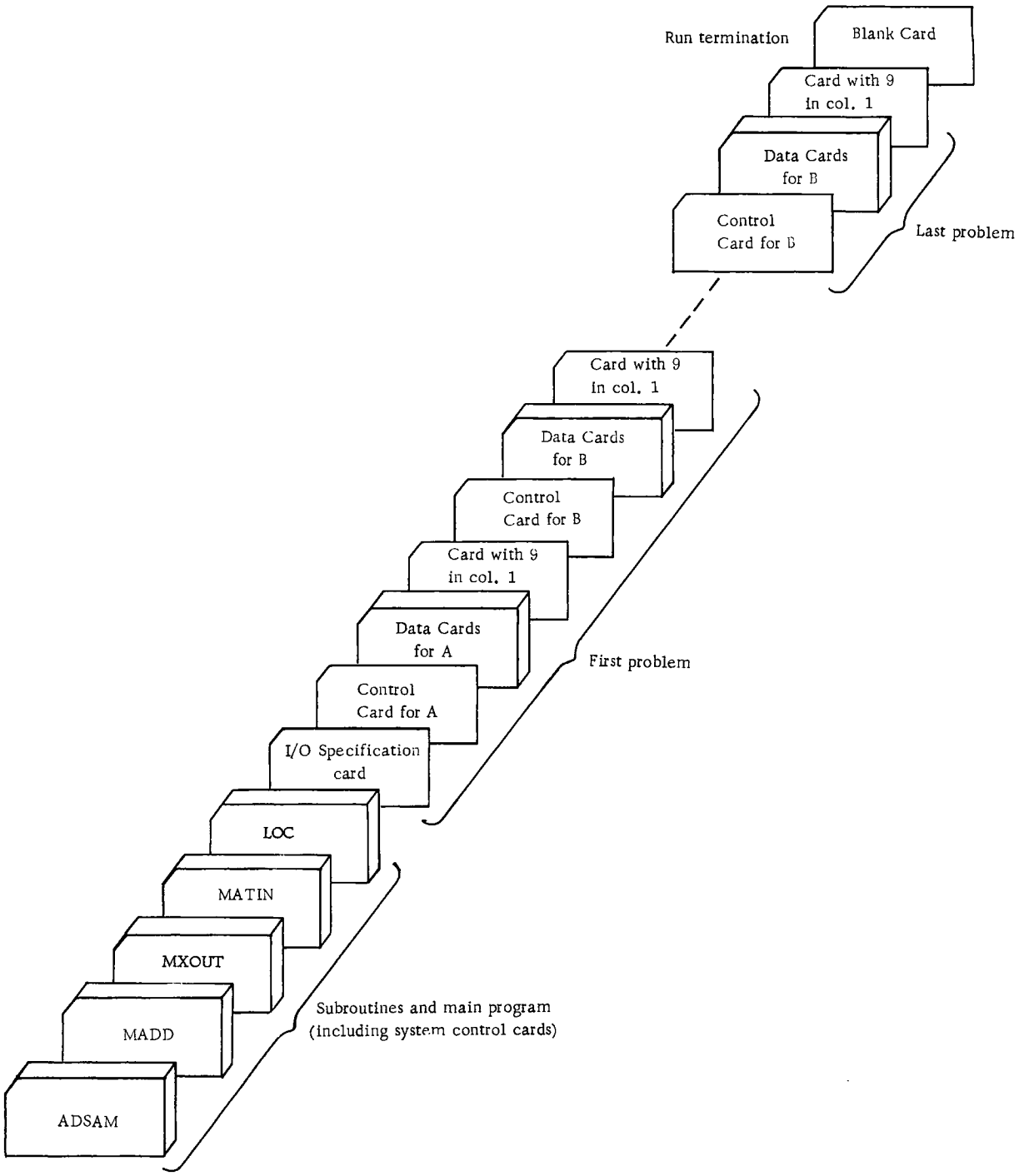


Figure 26. Deck setup (matrix addition)

program may be increased or decreased by making the following changes in ADSAM:

1. Modify the DIMENSION statement to reflect the number of elements for A, B, and R.
2. Insert the same number in the third parameter of the two CALL MATIN statements (20 and 45).

The output listing is set for 120 print positions across the page and double spacing. This can be changed by means of the last two arguments in the three CALL MXOUT statements in ADSAM (statements 40, 80, 90).

Operating Instructions

The matrix addition sample program is a standard FORTRAN program. Special operating instructions

are not required. Logical unit 2 is used for input, and logical unit 1 is used for output.

Error Messages

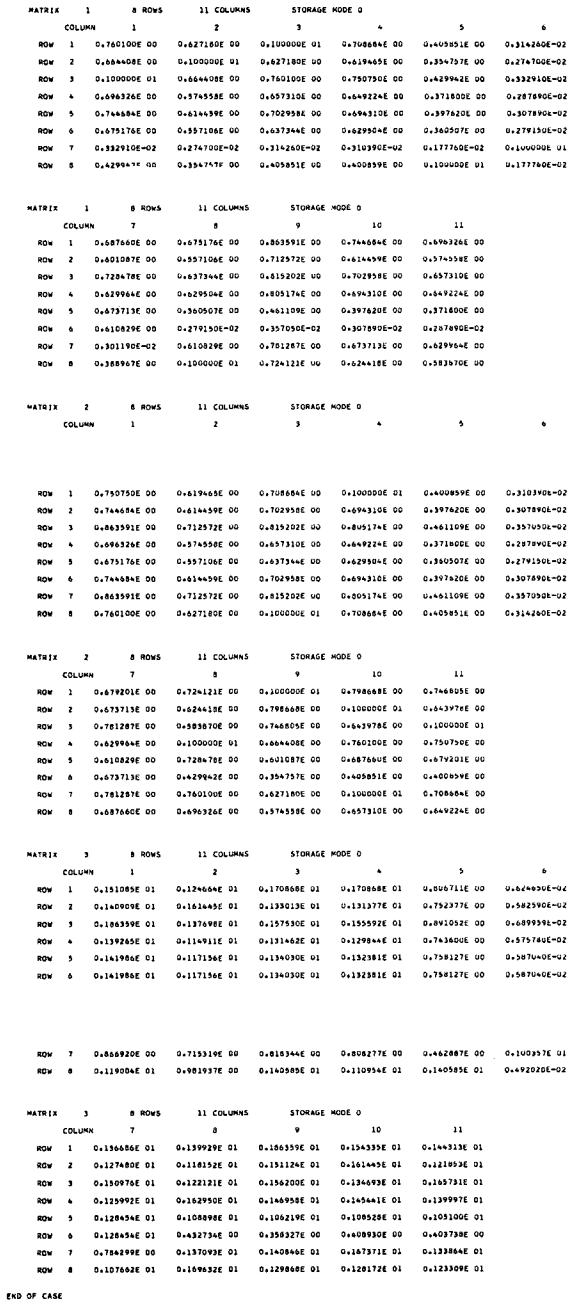
The following error conditions will result in messages:

1. Reserved storage area is too small for matrix: DIMENSIONED AREA TOO SMALL FOR INPUT MATRIX (matrix code no.). GO ON TO NEXT CASE.

2. Input matrices do not have the same dimensions: MATRIX DIMENSIONS NOT CONSISTENT. GO ON TO NEXT CASE.

3. Number of data cards does not correspond to that required by parameter card: INCORRECT NUMBER OF DATA CARDS FOR MATRIX (matrix code no.). EXECUTION TERMINATED.

Error conditions 1 and 2 allow the computer run to continue. Error condition 3, however, terminates execution and requires another run to process succeeding cases.



```
MATRIX    1       8 ROWS      11 COLUMNS      STORAGE MODE 0

          COLUMN      1             2             3             4             5             6
   ROW    1    0.760100E 00  0.627180E 00  0.100000E 01  0.708684E 00  0.405851E 00  0.314260E-02
   ROW    2    0.664408E 00  0.100000E 01  0.627180E 00  0.619465E 00  0.354757E 00  0.274700E-02
   ROW    3    0.100000E 01  0.664408E 00  0.760100E 00  0.750750E 00  0.429942E 00  0.332910E-02
   ROW    4    0.696326E 00  0.574558E 00  0.657310E 00  0.649224E 00  0.371800E 00  0.287690E-02
   ROW    5    0.744684E 00  0.614459E 00  0.702958E 00  0.694310E 00  0.397620E 00  0.307890E-02
   ROW    6    0.675176E 00  0.557106E 00  0.637344E 00  0.629504E 00  0.360507E 00  0.279150E-02
   ROW    7    0.332910E-02  0.274700E-02  0.314260E-02  0.310390E-02  0.177760E-02  0.100000E 01
   ROW    8    0.429942E 00  0.354757E 00  0.405851E 00  0.400859E 00  0.100000E 01  0.177760E-02

MATRIX    1       8 ROWS      11 COLUMNS      STORAGE MODE 0

          COLUMN      7             8             9             10            11
   ROW    1    0.687660E 00  0.675176E 00  0.863591E 00  0.744684E 00  0.696326E 00
   ROW    2    0.601087E 00  0.557106E 00  0.712572E 00  0.614459E 00  0.574558E 00
   ROW    3    0.728478E 00  0.637344E 00  0.815202E 00  0.702958E 00  0.657310E 00
   ROW    4    0.629984E 00  0.629504E 00  0.805174E 00  0.694310E 00  0.649224E 00
   ROW    5    0.673713E 00  0.360507E 00  0.461109E 00  0.397620E 00  0.371800E 00
   ROW    6    0.610829E 00  0.279150E-02  0.357050E-02  0.307890E-02  0.287690E-02
   ROW    7    0.301190E-02  0.610829E 00  0.781287E 00  0.673713E 00  0.629984E 00
   ROW    8    0.388967E 00  0.100000E 01  0.724121E 00  0.624418E 00  0.583870E 00

MATRIX    2       6 ROWS      11 COLUMNS      STORAGE MODE 0

          COLUMN      1             2             3             4             5             6
   ROW    1    0.750750E 00  0.619465E 00  0.708684E 00  0.100000E 01  0.400859E 00  0.310390E-02
   ROW    2    0.744684E 00  0.614459E 00  0.702958E 00  0.694310E 00  0.397620E 00  0.307890E-02
   ROW    3    0.863591E 00  0.712572E 00  0.815202E 00  0.805174E 00  0.461109E 00  0.357050E-02
   ROW    4    0.696326E 00  0.574558E 00  0.657310E 00  0.649224E 00  0.371800E 00  0.287690E-02
   ROW    5    0.675176E 00  0.557106E 00  0.637344E 00  0.629504E 00  0.360507E 00  0.279150E-02
   ROW    6    0.744684E 00  0.614459E 00  0.702958E 00  0.694310E 00  0.397620E 00  0.307890E-02
   ROW    7    0.863591E 00  0.712572E 00  0.815202E 00  0.805174E 00  0.461109E 00  0.357050E-02
   ROW    8    0.760100E 00  0.627180E 00  0.100000E 01  0.708684E 00  0.405851E 00  0.314260E-02

MATRIX    2       8 ROWS      11 COLUMNS      STORAGE MODE 0

          COLUMN      7             8             9             10            11
   ROW    1    0.679201E 00  0.724121E 00  0.100000E 01  0.798868E 00  0.746805E 00
   ROW    2    0.673713E 00  0.624418E 00  0.798868E 00  0.100000E 01  0.643978E 00
   ROW    3    0.781287E 00  0.583870E 00  0.746805E 00  0.643978E 00  0.100000E 01
   ROW    4    0.629984E 00  0.100000E 01  0.664408E 00  0.760100E 00  0.750750E 00
   ROW    5    0.610829E 00  0.728478E 00  0.601087E 00  0.687660E 00  0.679201E 00
   ROW    6    0.673713E 00  0.429942E 00  0.354757E 00  0.405851E 00  0.400859E 00
   ROW    7    0.781287E 00  0.760100E 00  0.627180E 00  0.100000E 01  0.708684E 00
   ROW    8    0.687660E 00  0.696326E 00  0.574558E 00  0.657310E 00  0.649224E 00

MATRIX    3       8 ROWS      11 COLUMNS      STORAGE MODE 0

          COLUMN      1             2             3             4             5             6
   ROW    1    0.151085E 01  0.124664E 01  0.170868E 01  0.170868E 01  0.806711E 00  0.624630E-02
   ROW    2    0.140909E 01  0.161445E 01  0.133013E 01  0.131377E 01  0.752377E 00  0.582590E-02
   ROW    3    0.186359E 01  0.137698E 01  0.157530E 01  0.155592E 01  0.891052E 00  0.689959E-02
   ROW    4    0.139265E 01  0.114911E 01  0.131462E 01  0.129844E 01  0.743600E 00  0.575784E-02
   ROW    5    0.141986E 01  0.117156E 01  0.134030E 01  0.132381E 01  0.758127E 00  0.587040E-02
   ROW    6    0.141986E 01  0.117156E 01  0.134030E 01  0.132381E 01  0.758127E 00  0.587040E-02

   ROW    7    0.886920E 00  0.715319E 00  0.818344E 00  0.808277E 00  0.462887E 00  0.100957E 01
   ROW    8    0.119084E 01  0.981937E 00  0.140585E 01  0.110954E 01  0.140585E 01  0.492020E-02

MATRIX    3       8 ROWS      11 COLUMNS      STORAGE MODE 0

          COLUMN      7             8             9             10            11
   ROW    1    0.136686E 01  0.139929E 01  0.186359E 01  0.154335E 01  0.144313E 01
   ROW    2    0.127480E 01  0.118152E 01  0.151124E 01  0.161445E 01  0.121853E 01
   ROW    3    0.150976E 01  0.122121E 01  0.156200E 01  0.136493E 01  0.165731E 01
   ROW    4    0.125992E 01  0.162950E 01  0.146958E 01  0.145441E 01  0.139997E 01
   ROW    5    0.128454E 01  0.108898E 01  0.106219E 01  0.108528E 01  0.105100E 01
   ROW    6    0.128454E 01  0.432734E 00  0.358327E 00  0.608930E 00  0.403738E 00
   ROW    7    0.786299E 00  0.137093E 01  0.140846E 01  0.167371E 01  0.133864E 01
   ROW    8    0.107662E 01  0.169832E 01  0.129868E 01  0.128172E 01  0.123309E 01

END OF CASE
```

Figure 27. Output listing

Sample Main Program for Matrix Addition - ADSAM

Purpose:

Matrix addition sample program.

Remarks:

I/O specifications transmitted to subroutines by COMMON.

Input card:

Column 2 MX - Logical unit number for output.

Column 4 MY - Logical unit number for input.

Subroutines and function subprograms required:

MADD

MATIN

MXOUT

LOC

Method:

Two input matrices are read from the standard input device. They are added and the resultant matrix is listed on the standard output device. This can be repeated for any number of pairs of matrices until a blank card is encountered.

```
// FOR
*IOCS(CARD,TYPEWRITER,1132 PRINTER)
*ONE WORD INTEGERS
C         SAMPLE MAIN PROGRAM FOR MATRIX ADDITION - ADSAM                ADSAM  1
C         MATRICES ARE DIMENSIONED FOR 1000 ELEMENTS. THEREFORE, PRODUCT ADSAM  2
C         OF NUMBER OF ROWS BY NUMBER OF COLUMNS CANNOT EXCEED 1000.     ADSAM  3
      DIMENSION A(650),B(650),R(650)                                     ADSAM  4
      COMMON MX,MY                                                       ADSAM  5
   10 FORMAT(////16H MATRIX ADDITION)                                    ADSAM  6
   11 FORMAT(//45H DIMENSIONED AREA TOO SMALL FOR INPUT MATRIX ,I4)      ADSAM  7
   12 FORMAT(//21H EXECUTION TERMINATED)                                 ADSAM  8
   13 FORMAT(//33H MATRIX DIMENSIONS NOT CONSISTENT)                     ADSAM  9
   14 FORMAT(//43H INCORRECT NUMBER OF DATA CARDS FOR MATRIX ,I4)        ADSAM 10
   15 FORMAT(//19H GO ON TO NEXT CASE)                                   ADSAM 11
   16 FORMAT(//12H END OF CASE)                                          ADSAM 12
   17 FORMAT(2I2)                                                        ADSAM 13
      READ(2,17)MX,MY                                                    ADSAM 14
      WRITE(MX,10)                                                       ADSAM 15
   20 CALL MATIN(ICODA,A, 100,NA,MA,MSA,IER)                             ADSAM 16
      IF( NA ) 25,95,25                                                  ADSAM 17
   25 IF(IER-1) 40,30,35                                                 ADSAM 18
   30 WRITE(MX,11) ICODA                                                 ADSAM 19
      GO TO 45                                                           ADSAM 20
   35 WRITE(MX,14) ICODA                                                 ADSAM 21
   37 WRITE(MX,12)                                                       ADSAM 22
      GO TO 95                                                           ADSAM 23
   40 CALL MXOUT(ICODA,A,NA,MA,MSA,60,120,2)                             ADSAM 24
   45 CALL MATIN(ICODB,B, 100,NB,MB,MSB,IER)                             ADSAM 25
      IF(IER-1) 60,50,55                                                 ADSAM 26
   50 WRITE(MX,11)ICODB                                                  ADSAM 27
      WRITE(MX,15)                                                       ADSAM 28
      GO TO 20                                                           ADSAM 29
   55 WRITE(MX,14) ICODB                                                 ADSAM 30
      GO TO 37                                                           ADSAM 31
   60 IF(NA-NB) 75,70,75                                                 ADSAM 32
   70 IF(MA-MB) 75,80,75                                                 ADSAM 33
   75 WRITE(MX,13)                                                       ADSAM 34
      WRITE(MX,15)                                                       ADSAM 35
      GO TO 20                                                           ADSAM 36
   80 CALL MXOUT(ICODB,B,NB,MB,MSB,60,120,2)                             ADSAM 37
      ICODR=ICODA+ICODB                                                  ADSAM 38
      CALL MADD(A,B,R,NA,MA,MSA,MSB)                                     ADSAM 39
      MSR=MSA                                                            ADSAM 40
      IF(MSA-MSB) 90,90,85                                               ADSAM 41
   85 MSR=MSB                                                            ADSAM 42
   90 CALL MXOUT(ICODR,R,NA,MA,MSR,60,120,2)                             ADSAM 43
      WRITE(MX,16)                                                       ADSAM 44
      GO TO 20                                                           ADSAM 45
   95 STOP                                                               ADSAM 46
      END                                                                ADSAM 47
// DUP
*STORE      WS  UA  ADSAM
// XEQ ADSAM
```

```
1 2                                                                      1
 00010008001100                                                          2
0.7601008 0.6271802 1.0000000 0.7086843 0.4058519 0.0031426 0.6876602    3
0.6751766 0.8635910 0.7446845 0.6963269                                  4
0.6644085 1.0000000 0.6271802 0.6194650 0.3547574 0.0027470 0.6010878    5
0.5571068 0.7125728 0.6144597 0.5745585                                  6
1.0000000 0.6644085 0.7601008 0.7507505 0.4299425 0.0033291 0.7284786    7
0.6373449 0.8152021 0.7029582 0.6573101                                  8
0.6963269 0.5745585 0.6573101 0.6492243 0.3718001 0.0028789 0.6299642    9
0.6295047 0.8051740 0.6943108 0.6492243                                 10
0.7446845 0.6144597 0.7029582 0.6943108 0.3976204 0.0030789 0.6737132    11
0.3605070 0.4611099 0.3976204 0.3718001                                 12
0.6751766 0.5571068 0.6373449 0.6295047 0.3605070 0.0027915 0.6108296    13
0.0027915 0.0035705 0.0030789 0.0028789                                 14
0.0033291 0.0027470 0.0031426 0.0031039 0.0017776 1.0000000 0.0030119    15
0.6108296 0.7812874 0.6737132 0.6299642                                 16
0.4299425 0.3547574 0.4058519 0.4008593 1.0000000 0.0017776 0.3889673    17
1.0000000 0.7241215 0.6244183 0.5838704                                 18
9                                                                       19
 00020008001100                                                         20
0.7507505 0.6194650 0.7086843 1.0000000 0.4008593 0.0031039 0.6792011    21
0.7241215 1.0000000 0.7986682 0.7468050                                 22
0.7446845 0.6144597 0.7029582 0.6943108 0.3976204 0.0030789 0.6737132    23
0.6244183 0.7986682 1.0000000 0.6439786                                 24
0.8635910 0.7125728 0.8152021 0.8051740 0.4611099 0.0035705 0.7812874    25
0.5838704 0.7468050 0.6439786 1.0000000                                 26
0.6963269 0.5745585 0.6573101 0.6492243 0.3718001 0.0028789 0.6299642    27
1.0000000 0.6644085 0.7601008 0.7507505 0.4299425 0.0033291 0.7284786    28
0.6751766 0.5571068 0.6373449 0.6295047 0.3605070 0.0027915 0.6108296    29
0.7284786 0.6010878 0.6876602 0.6792011 0.3889673 0.0030119 1.0000000    30
0.7446845 0.6144597 0.7029582 0.6943108 0.3976204 0.0030789 0.6737132    31
0.4299425 0.3547574 0.4058519 0.4008593 1.0000000 0.0017776 0.3889673    32
0.8635910 0.7125728 0.8152021 0.8051740 0.4611099 0.0035705 0.7812874    33
0.7601008 0.6271802 1.0000000 0.7086843 0.4058519 0.0031426 0.6876602    34
0.7601008 0.6271802 1.0000000 0.7086843 0.4058519 0.0031426 0.6876602    35
0.6963269 0.5745585 0.6573101 0.6492243 0.3718001 0.0028789 0.6299642    36
9                                                                       37
                                                                        38
```

```
   SUBROUTINE MATIN

   PURPOSE
      READS CONTROL CARD AND MATRIX DATA ELEMENTS FROM LOGICAL
      UNIT 5

   USAGE
      CALL MATIN(ICODE,A,ISIZE,IROW,ICOL,IS,IER)

   DESCRIPTION OF PARAMETERS
      ICODE-UPON RETURN, ICODE WILL CONTAIN FOUR DIGIT
            IDENTIFICATION CODE FROM MATRIX PARAMETER CARD
      A     -DATA AREA FOR INPUT MATRIX
      ISIZE-NUMBER OF ELEMENTS DIMENSIONED BY USER FOR AREA A
      IROW -UPON RETURN, IROW WILL CONTAIN ROW DIMENSION FROM
            MATRIX PARAMETER CARD
      ICOL -UPON RETURN, ICOL WILL CONTAIN COLUMN DIMENSION FROM
            MATRIX PARAMETER CARD
      IS   -UPON RETURN, IS WILL CONTAIN STORAGE MODE CODE FROM
            MATRIX PARAMETER CARD WHERE
            IS=0 GENERAL MATRIX
            IS=1 SYMMETRIC MATRIX
            IS=2 DIAGONAL MATRIX
      IER  -UPON RETURN, IER WILL CONTAIN AN ERROR CODE WHERE
            IER=0    NO ERROR
            IER=1    ISIZE IS LESS THAN NUMBER OF ELEMENTS IN
                     INPUT MATRIX
            IER=2    INCORRECT NUMBER OF DATA CARDS

   REMARKS
      NONE

   SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
      LOC

   METHOD
      SUBROUTINE ASSUMES THAT INPUT MATRIX CONSISTS OF PARAMETER
      CARD FOLLOWED BY DATA CARDS
      PARAMETER CARD HAS THE FOLLOWING FORMAT
        COL. 1- 2 BLANK
        COL. 3- 6 UP TO FOUR DIGIT IDENTIFICATION CODE
        COL. 7-10 NUMBER OF ROWS IN MATRIX
        COL.11-14 NUMBER OF COLUMNS IN MATRIX
        COL.15-16 STORAGE MODE OF MATRIX WHERE
            0 - GENERAL MATRIX
            1 - SYMMETRIC MATRIX
            2 - DIAGONAL MATRIX
      DATA CARDS ARE ASSUMED TO HAVE SEVEN FIELDS OF TEN COLUMNS
      EACH.  DECIMAL POINT MAY APPEAR ANYWHERE IN A FIELD.  IF NO
      DECIMAL POINT IS INCLUDED, IT IS ASSUMED THAT THE DECIMAL
      POINT IS AT THE END OF THE 10 COLUMN FIELD. NUMBER IN EACH
      FIELD MAY BE PRECEDED BY BLANKS.  DATA ELEMENTS MUST BE
      PUNCHED BY ROW.  A ROW MAY CONTINUE FROM CARD TO CARD.
      HOWEVER EACH NEW ROW MUST START IN THE FIRST FIELD OF THE
      NEXT CARD.  ONLY THE UPPER TRIANGULAR PORTION OF A SYMMETRIC
      OR THE DIAGONAL ELEMENTS OF A DIAGONAL MATRIX ARE CONTAINED
      ON DATA CARDS.  THE FIRST ELEMENT OF EACH NEW ROW WILL BE
      THE DIAGONAL ELEMENT FOR A MATRIX WITH  SYMMETRIC OR
      DIAGONAL STORAGE MODE. COLUMNS 71-80 OF DATA CARDS MAY BE
      USED FOR IDENTIFICATION, SEQUENCE NUMBERING, ETC..
      THE LAST DATA CARD FOR ANY MATRIX MUST BE FOLLOWED BY A CARD
      WITH A 9 PUNCH IN COLUMN 1.
```

```
      SUBROUTINE MATIN(ICODE,   A,ISIZE,IROW,ICOL,IS,IER)                MATIN  1
      DIMENSION A(1)                                                     MATIN  2
      DIMENSION CARD(8)                                                  MATIN  3
      COMMON MX,MY                                                       MATIN  4
    1 FORMAT(7F10.0)                                                     MATIN  5
    2 FORMAT(I6,2I4,I2)                                                  MATIN  6
    3 FORMAT(11)                                                         MATIN  7
      IUC=7                                                              MATIN  8
      IER=0                                                              MATIN  9
      READ( MY,2)ICODE,IROW,ICOL,IS                                      MATIN 10
      CALL LOC(IROW,ICOL,ICNT,IROW,ICOL,IS)                              MATIN 11
      IF(ISIZE-ICNT)6,7,7                                                MATIN 12
    6 IER=1                                                              MATIN 13
    7 IF (ICNT)38,38,8                                                   MATIN 14
    8 ICOLT=ICOL                                                         MATIN 15
      IROCR=1                                                            MATIN 16
C       COMPUTE NUMBER OF CARDS FOR THIS ROW                             MATIN 17
   11 IRCDS=(ICOLT-1)/IUC+1                                              MATIN 18
      IF(IS-1)15,15,12                                                   MATIN 19
   12 IRCDS=1                                                            MATIN 20
C       SET UP LOOP FOR NUMBER OF CARDS IN ROW                           MATIN 21
   15 DO 31 K=1,IRCDS                                                    MATIN 22
      READ(MY,1)(CARD(I),I=1,IUC)                                        MATIN 23
C       SKIP THROUGH DATA CARDS IF INPUT AREA TOO SMALL                  MATIN 24
      IF(IER)16,16,31                                                    MATIN 25
   16 L=0                                                                MATIN 26
C       COMPUTE COLUMN NUMBER FOR FIRST FIELD IN CURRENT CARD            MATIN 27
      JS=(K-1)*IUC+ICOL-ICOLT+1                                          MATIN 28
```

```
   SUBROUTINE MXOUT

   PURPOSE
      PRODUCES AN OUTPUT LISTING OF ANY SIZED ARRAY ON
      LOGICAL UNIT 3
```

```
USAGE
   CALL MXOUT(ICODE,A,N,M,MS,LINS,IPOS,ISP)

DESCRIPTION OF PARAMETERS
   ICODE- INPUT CODE NUMBER TO BE PRINTED ON EACH OUTPUT PAGE
   A-NAME OF OUTPUT MATRIX
   N-NUMBER OF ROWS IN A
   M-NUMBER OF COLUMNS IN A
   MS-STORAGE MODE OF A WHERE MS=
         0-GENERAL
         1-SYMMETRIC
         2-DIAGONAL
   LINS-NUMBER OF PRINT LINES ON THE PAGE (USUALLY 60)
   IPOS-NUMBER OF PRINT POSITIONS ACROSS THE PAGE (USUALLY 132)
   ISP-LINE SPACING CODE, 1 FOR SINGLE SPACE, 2 FOR DOUBLE
       SPACE

REMARKS
   NONE

SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
   LOC

METHOD
   THIS SUBROUTINE CREATES A STANDARD OUTPUT LISTING OF ANY
   SIZED ARRAY WITH ANY STORAGE MODE. EACH PAGE IS HEADED WITH
   THE CODE NUMBER,DIMENSIONS AND STORAGE MODE OF THE ARRAY.
   EACH COLUMN AND ROW IS ALSO HEADED WITH ITS RESPECTIVE
   NUMBER.
```

```
      SUBROUTINE MXOUT (ICODE,A,N,M,MS,LINS,IPOS,ISP)                    MXOUT  1
      DIMENSION A(1),B(8)                                                MXOUT  2
      COMMON MX,MY                                                       MXOUT  3
    1 FORMAT(////5X, 7HMATRIX ,I5,6X,I3,5H ROWS,6X,I3,8H COLUMNS,        MXOUT  4
     18X,13HSTORAGE MODE ,I1,/)                                          MXOUT  5
    2 FORMAT(12X,8HCOLUMN  ,7(3X,I3,10X)//)                            MXOUT  6
    4 FORMAT(7X,4HROW ,I3,7(E16.6))                                    MXOUT  7
    5 FORMAT(/,7X,4HROW ,I3,7(E16.6))                                  MXOUT  8
      J=1                                                              MXOUT  9
C        WRITE HEADING                                                 MXOUT 10
      NEND=IPOS/16-1                                                   MXOUT 11
      LEND = (LINS/ISP)-10                                             MXOUT 12
   10 LSTRT=1                                                          MXOUT 13
   20 WRITE(MX,1)ICODE,N,M,MS                                          MXOUT 14
      JNT=J+NEND-1                                                     MXOUT 15
      IF(JNT-M)33,33,32                                                MXOUT 16
   32 JNT=M                                                            MXOUT 17
   33 CONTINUE                                                         MXOUT 18
      WRITE(MX,2)(JCUR,JCUR=J,JNT)                                     MXOUT 19
      LTEND = LSTRT+LEND-1                                             MXOUT 20
      DO 80 L=LSTRT,LTEND                                              MXOUT 21
C        FORM OUTPUT ROW LINE                                          MXOUT 22
      DO 55 K=1,NFND                                                   MXOUT 23
      KK=K                                                             MXOUT 24
      JT = J+K-1                                                       MXOUT 25
      CALL LOC(L,JT,IJYT,N,M,MS)                                       MXOUT 26
      B(K)=0.0                                                         MXOUT 27
      IF(IJNT)50,50,45                                                 MXOUT 28
   45 B(K)=A(IJNT)                                                     MXOUT 29
   50 CONTINUE                                                         MXOUT 30
C        CHECK IF LAST COLUMN.  IF YES GO TO 60                        MXOUT 31
      IF(JT-M) 55,60,60                                                MXOUT 32
   55 CONTINUE                                                         MXOUT 33
C        END OF LINE, NOW WRITE                                        MXOUT 34
   60 IF(ISP-1)65,65,70                                                MXOUT 35
   65 WRITE(MX,4)L,(B(JW),JW=1,KK)                                     MXOUT 36
      GO TO 75                                                         MXOUT 37
   70 WRITE(MX,5)L,(B(JW),JW=1,KK)                                     MXOUT 38
C        IF END OF ROWS,GO CHECK COLUMNS                               MXOUT 39
   75 IF(N-L)85,85,80                                                  MXOUT 40
   80 CONTINUE                                                         MXOUT 41
C        WRITE NEW HEADING                                             MXOUT 42
      LSTRT=LSTRT+LEND                                                 MXOUT 43
      GO TO 20                                                         MXOUT 44
C        END OF COLUMNS, THEN RETURN                                   MXOUT 45
   85 IF(JT-M)90,95,95                                                 MXOUT 46
   90 J=JT+1                                                           MXOUT 47
      GO TO 10                                                         MXOUT 48
   95 RETURN                                                           MXOUT 49
      END                                                              MXOUT 50
```

NUMERICAL QUADRATURE INTEGRATION

Problem Description

The tabulated values of a function for a given spacing are integrated. Multiple sets of tabulated values may be processed.

Program

Description

The numerical quadrature integration program consists of a main routine QDINT, and one subroutine, QSF, from the Scientific Subroutine Package.

Capacity

The capacity of the sample program and the format for data input have been set up as follows:

1. Up to 500 tabulated values of a function
2. (7F10.0) format for input data cards

Therefore, if the problem satisfies the above conditions, no modification to the sample program is necessary. However, if there are more than 500 values to be integrated the dimension statement in the sample main program must be modified to handle this particular problem. Similarly, if input data cards are prepared using a different format, the input format statement in the sample main program must be modified. The general rules for program modification are described later.

Input

I/O Specification Card

Each integration requires a parameter card with the following format:

Columns	Contents	For Sample Problem
1 - 5	Up to 5-digit numeric identification code	12345
6 - 10	Number of tabulated values for this function	0020
11 - 20	Interval between tabulated values	1.0

The first two parameters consist of up to five digits with no decimal point (FORMAT (215)). Note that the second parameter may not exceed 500. The third parameter consists of up to ten digits (FORMAT) (F10.0).

Data Cards

Data cards are assumed to be seven fields of ten columns each. The decimal point may appear anywhere in the field, or be omitted, but the number must be right-justified. The number in each field may be preceded by blanks. Columns 71 through 80 of the data cards may be used for identification, sequence numbering, etc. If there are more than seven tabulated values, the values should continue from card to card with seven values per card, until the number of values specified in the parameter card has been reached.

A blank card following the last set of data terminates the run.

Deck Setup

The deck setup is shown in Figure 28.

Sample

A listing of input cards for the sample problem is presented at the end of the sample main program.

Output

Description

The identification code number, number of tabulated input values, the interval for the tabulated values, and the resultant integral values at each step are printed.

Sample

The output listing for the sample problem is shown in Figure 29.

Program Modification

Noting that storage problems may result, as previously discussed in "Sample Program Description", the maximum number of tabulated values acceptable to the sample program may be increased. Input data in a different format can also be handled by providing a specific format statement.

1. Modify the DIMENSION statement in QDINT so that the size of array Z is equal to the maximum number of tabulated values.
2. Changes to the format of the parameter cards and data cards may be made by modifying FORMAT statements 10 and 32, respectively, in QDINT.

```
INTEGRATION OF TABULATED VALUES FOR DY/DX USING SUBROUTINE QSF

FUNCTION  12345      20 TABULATED VALUES      INTERVAL = 0.10000002E 01

     RESULTANT VALUE OF INTEGRAL AT EACH STEP IS
0.00000000E 00 0.19999983E 01 0.39999995E 01 0.59999981E 01 0.79999990E 01 0.99999981E 01
0.11999998E 02 0.13999996E 02 0.15999996E 02 0.17999996E 02 0.19999996E 02 0.21999992E 02
0.23999992E 02 0.25999988E 02 0.27999988E 02 0.29999984E 02 0.31999984E 02 0.33999984E 02
0.35999984E 02 0.37999977E 02


INTEGRATION OF TABULATED VALUES FOR DY/DX USING SUBROUTINE QSF

FUNCTION    543      10 TABULATED VALUES      INTERVAL = 0.10000002E 01

     RESULTANT VALUE OF INTEGRAL AT EACH STEP IS
0.00000000E 00 0.14999959E 01 0.39999995E 01 0.74999952E 01 0.11999998E 02 0.17499996E 02
0.23999996E 02 0.31499992E 02 0.39999992E 02 0.49499984E 02
```

Figure 29. Output listing

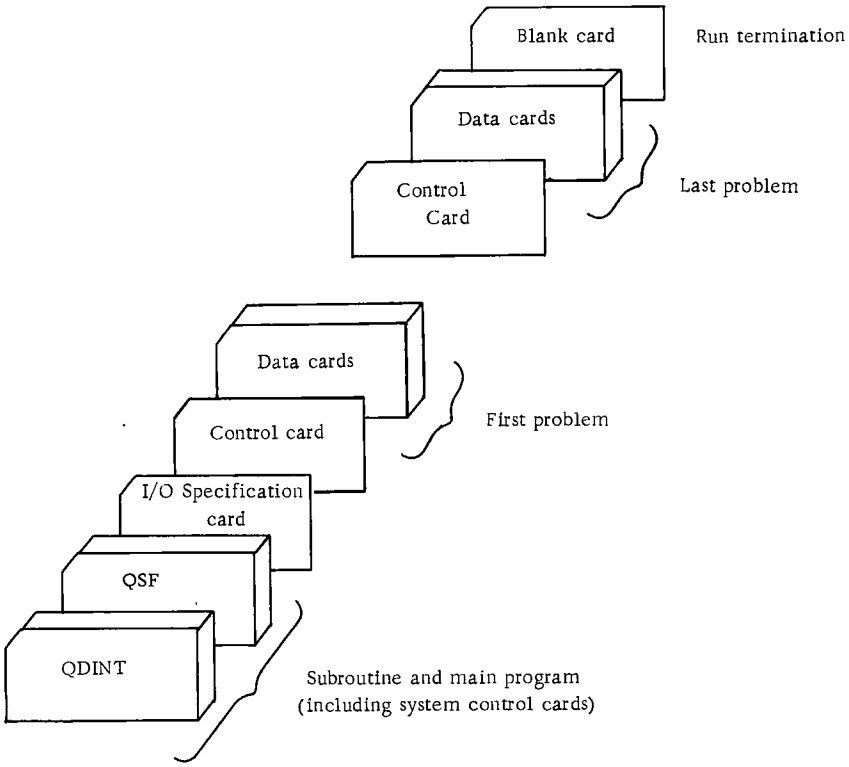


Figure 28. Deck setup (numerical quadrature integration)

Operating Instructions

The numerical quadrature integration sample program is a standard FORTRAN program. Special operating instructions are not required. Logical unit 2 is used for input, and logical unit 1 is used for output.

Error Messages

The following conditions will result in error messages:

1. The number of tabulated values specified in the parameter card is less than or equal to two: ILLEGAL CONDITION. NUMBER OF TABULATED VALUES IS LESS THAN THREE.

The program will continue to read data cards until the next problem is reached.

2. The interval specified in the parameter card is zero: ILLEGAL CONDITION. SPECIFIED INTERVAL IS ZERO.

The program will continue to read data cards until the next problem is reached.

Sample Program for Integration of a Tabulated Function by Numerical Quadrature - QDINT

Purpose:

Integrates a set of tabulated values for F(X) given the number of values and their spacing.

Remarks:

The number of values must be more than two and the spacing greater than zero.
I/O logical units determined by MX and MY, respectively.

Subroutines and function subprograms required:

QSF

Method:

Reads control card containing the code number, number of values, and the spacing of the function values contained on the following data cards. Data cards are then read and integration is performed. More than one control card and corresponding data can be integrated in one run. Execution is terminated by a blank control card.

```
// FOR
*IOCS(CARD,TYPEWRITER,1132 PRINTER)
*ONE WORD INTEGERS
C        SAMPLE PROGRAM FOR INTEGRATION OF A TABULATED FUNCTION BY      QDINT  1
C        NUMERICAL QUADRATURE - QDINT                                   QDINT  2
C        THE FOLLOWING DIMENSION MUST BE AS LARGE AS THE MAXIMUM NUMBER QDINT  3
C        OF TABULATED VALUES TO BE INTEGRATED                           QDINT  4
      DIMENSION Z(500)                                                  QDINT  5
   10 FORMAT (2I5,F10.0)                                                QDINT  6
   20 FORMAT(///////' INTEGRATION OF TABULATED VALUES FOR DY/DX USING SUBQDINTM01
     1ROUTINE QSF' ///11H FUNCTION  ,I5,3X,I5,17H TABULATED VALUES,     QDINTM02
     25X,10HINTERVAL =,E15.8/)                                          QDINTM03
   22 FORMAT(/18H ILLEGAL CONDITION/)                                   QDINT 10
   23 FORMAT(/46H NUMBER OF TABULATED VALUES IS LESS THAN THREE)        QDINTM04
   24 FORMAT(/27H SPECIFIED INTERVAL IS ZERO)                           QDINT 12
   30 FORMAT(//7X,  'RESULTANT VALUE OF INTEGRAL AT EACH STEP IS',//    QDINTM05
     1(1H ,6E15.8))                                                     QDINTM06
   31 FORMAT(2I2)                                                       QDINT 14
   32 FORMAT(7F10.0)                                                    QDINT 15
      READ(2,31)MX,MY                                                   QDINT 16
   35 READ(MY,10)ICOD,NUMBR,SPACE                                       QDINT 17
      IF(ICOD+NUMBR)70,70,38                                            QDINT 18
   70 STOP                                                              QDINT 19
   38 WRITE(MX,20)ICOD,NUMBR,SPACE                                      QDINT 20
      IF(NUMBR-3)100,50,50                                              QDINTM07
   50 READ(MY,32)(Z(I),I=1,NUMBR)                                       QDINT 22
      CALL QSF (SPACE,Z,Z,NUMBR)                                        QDINTM08
      IF(SPACE)60,200,60                                                QDINTM09
   60 WRITE(MX,30)(Z(I),I=1,NUMBR)                                      QDINTM10
      GO TO 35                                                          QDINT 26
  100 WRITE(MX,22)                                                      QDINT 27
      WRITE(MX,23)                                                      QDINT 28
      READ(MY,32)(Z(I),I=1,NUMBR)                                       QDINTM11
      GO TO 35                                                          QDINT 29
  200 WRITE(MX,22)                                                      QDINT 30
      WRITE(MX,24)                                                      QDINT 31
      GO TO 35                                                          QDINT 32
      END                                                               QDINT 33
// DUP
*STORE      WS  UA  QDINT
// XEQ QDINT
```

```
 1 2                                                                    1
12345   20      1.0                                                    2
        2.0     2.0     2.0     2.0     2.0     2.0     2.0            3
        2.0     2.0     2.0     2.0     2.0     2.0     2.0            4
        2.0     2.0     2.0     2.0     2.0     2.0                    5
  543   10      1.0                                                    6
        1.0     2.0     3.0     4.0     5.0     6.0     7.0            7
        8.0     9.0    10.0                                            8
                                                                       9
```

RUNGE-KUTTA INTEGRATION

Problem Description

A differential equation of the form:

$$\frac{dy}{dx} = f(x, y)$$

is integrated with initial conditions as specified in a parameter card. The differential equation is defined in the form of a function subprogram that is provided by the user.

Program

Description

The Runge-Kutta integration program consists of a main routine, RKINT, one subroutine, RK2, from the Scientific Subroutine Package, and one user-supplied function subprogram, FUN, which defines the differential equation to be integrated.

Capacity

Up to 500 values of the integral may be tabulated.

Input

I/O Specification Card

Each integration requires a control card with the following format:

Columns	Contents	For Sample Problem
1 - 10	Initial value of $X = X_0$	1.0
11 - 20	Initial value of $Y = Y(X_0)$	0.0

Columns	Contents	For Sample Problem
21 - 30	Step size	0.01
31 - 35	Number of steps required between tabulated values	10
36 - 40	Total number of tabulated values required	30

The first three parameters consist of up to ten digits.

(FORMAT (F10.0))

The last two parameters consist of up to four digits plus a blank.

(FORMAT (15))

Multiple parameter cards may be used.

A blank card terminates the run.

Data Cards

None.

Blank Card

Run termination.

Deck Setup

The deck setup is shown in Figure 30.

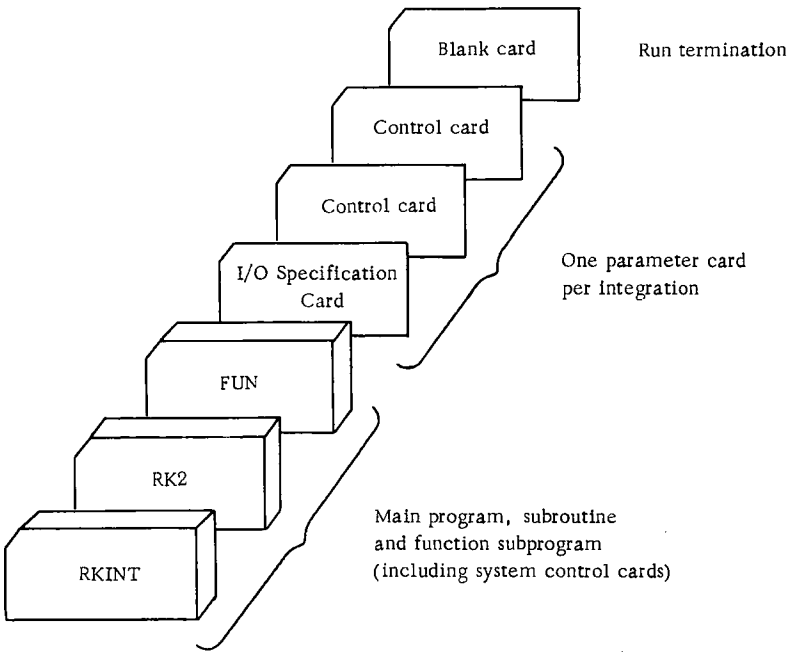


Figure 30. Deck setup (Runge-Kutta integration)

Sample

A listing of the input cards for the sample problem is presented at the end of the sample main program.

Output

Description

The values for the initial conditions and the tabulated values of the integral are printed.

Sample

The output listing for the sample problem is shown in Figure 31.

Program Modification

Noting that storage problems may result, as previously described in "Sample Program Description", the maximum number of tabulated values acceptable to the sample program may be increased. Input data in a different format can also be handled by providing a specific format statement.

1. Modify the DIMENSION statement in RKINT so that array A is as large as the number of tabulated values.

```
SOLUTION OF DY/DX=FUN(X,Y) BY RK2 SUBROUTINE

  H=  0.010  X0=  1.000  Y0=  0.000

    X                Y(X)

  1.10          0.95310136E-01
  1.20          0.18232139E 00
  1.30          0.26236397E 00
  1.40          0.33647167E 00
  1.50          0.40546423E 00
  1.60          0.47000247E 00
  1.70          0.53062677E 00
  1.80          0.58778464E 00
  1.90          0.64185142E 00
  2.00          0.69314432E 00
  2.10          0.74193394E 00
  2.20          0.78845346E 00
  2.30          0.83290457E 00
  2.40          0.87546372E 00
  2.50          0.91628539E 00
  2.60          0.95550584E 00
  2.70          0.99324584E 00
  2.80          0.10296125E 01
  2.90          0.10647029E 01
  3.00          0.10986037E 01
  3.10          0.11313924E 01
  3.20          0.11631403E 01
  3.30          0.11939110E 01
  3.40          0.12237627E 01
  3.50          0.12527496E 01
  3.60          0.12809195E 01
  3.70          0.13083176E 01
  3.80          0.13349850E 01
  3.90          0.13609600E 01
  4.00          0.13862772E 01
```

Figure 31. Output listing

2. Changes to the format of the parameter card may be made by modifying FORMAT statement 1.

The user-supplied function subprogram FUN may be replaced by any function subprogram having the same name and parameter list. In this way, the user may define any desired first-order differential equation.

Operating Instructions

The sample program for Runge-Kutta integration is a standard FORTRAN program. Special operating instructions are not required. Logical unit 2 is used for input, and logical unit 1 is used for output.

Error Messages

None.

Sample Program for Runge-Kutta Integration of a Given Function with Tabulated Output - RKINT

Purpose:

Integrates the function subprogram FUN using the initial conditions contained in control cards. Produces tabulated output.

Remarks:

I/O logical units determined by MX and MY, respectively.

Subroutines and function subprograms required:

RK2

FUN - User-supplied function subprogram giving DY/DX=FUN(X, Y)

Method:

Reads control card containing initial values of X and Y, step size, number of steps desired between tabulated values, and number of tabulated values required. Program then enters RK2 to perform integration. Multiple control cards can be used on the same function.

```
// FOR
*IOCS(CARD,TYPEWRITER,1132 PRINTER)
*ONE WORD INTEGERS
C        SAMPLE PROGRAM FOR RUNGE-KUTTA INTEGRATION OF A GIVEN FUNCTION RKINT  1
C        WITH TABULATED OUTPUT - RKINT                                  RKINT  2
      EXTERNAL FUN                                                      RKINT  3
C        THE FOLLOWING DIMENSION MUST BE AS LARGE AS THE MAXIMUM        RKINT  4
C        NUMBER OF TABULATED VALUES DESIRED                             RKINT  5
      DIMENSION A(500)                                                  RKINT  6
    1 FORMAT (3F10.0,2I5)                                               RKINT  7
    2 FORMAT(1////7X,44HSOLUTION OF DY/DX=FUN(X,Y) BY RK2 SUBROUTINE///, RKINT  8
     1 10X,2HH=,F7.3,2X,3HX0=,F7.3,2X,3HY0=,F7.3///12X,1HX,18X,4HY(X)//) RKINT  9
    3 FORMAT(/10X,F5.2,10X,E15.8)                                       RKINT 10
    4 FORMAT(2I2)                                                       RKINT 11
      READ(2,4)MX,MY                                                    RKINT 12
C        READ CONTROL CARD CONTAINING ITEMS LISTED UNDER METHOD.        RKINT 13
   10 READ(MY,1)X0,Y0,H,JNT,IENT                                        RKINT 14
C        CHECK IF CARD IS BLANK. IF SO, RETURN.                         RKINT 15
      IF(IENT)20,40,20                                                  RKINT 16
C        WRITE HEADING INFORMATION.                                     RKINT 17
   20 WRITE(MX,2)H,X0,Y0                                                RKINT 18
C        PERFORM INTEGRATION                                            RKINT 19
      CALL RK2(FUN,H,X0,Y0,JNT,IENT,A)                                  RKINT 20
C        WRITE OUTPUT                                                   RKINT 21
      STEP=FLOAT(JNT)*H                                                 RKINT 22
      X=X0                                                              RKINT 23
      DO 30 I=1,IENT                                                    RKINT 24
      X = X+STEP+.1E-05                                                 RKINT 25
   30 WRITE(MX,3)X,A(I)                                                 RKINT 26
C        GO BACK AND CHECK FOR ADDITIONAL CONTROL CARD.                 RKINT 27
      GO TO 10                                                          RKINT 28
   40 STOP                                                              RKINT 29
      END                                                               RKINT 30
// DUP
*STORE      WS  UA  RKINT
// XEQ RKINT
```

```
1 2
     1.0       0.0       .01   10   90
```

```
  FUNCTION FUN(X,Y)                                                 FUN   1
  FUN=1./X                                                          FUN   2
  RETURN                                                            FUN   3
  END                                                               FUN   4
```

POLYNOMIAL ROOTS

Problem Description

The real and complex roots are computed for a real polynomial with given coefficients. Multiple sets of coefficients may be processed.

Program

Description

The polynomial roots sample program consists of a main routine, SMPRT, and one subroutine, POLRT, from the Scientific Subroutine Package.

Capacity

Roots for polynomials of order 36 or less may be computed.

Input

I/O Specification Card

Each set of data requires a control card with the following format:

Columns	Contents	For Sample Problem
1	Blank	
2 - 5	Up to four-digit identification code	360
6 - 8	Blank	
9 - 10	Order of polynomial	9

The first parameter consists of up to four digits without decimal point (I4).

The second parameter consists of up to two digits with no decimal point (I2). The order of the polynomial must be less than or equal to 36.

Data Cards

Data cards are assumed to have seven fields of ten columns each. The decimal point may appear anywhere in the field, or be omitted, but the number must be right-justified. The number in each field may be preceded by blanks. Columns 71 to 80 of the data cards may be used for identification, sequence numbering, etc. If there are more than seven coefficients, the values should continue from card to card with seven values per card until the number of values has been reached that is one greater than the order of the polynomial. The first coefficient is for the constant term of the polynomial and the last coefficient for the highest order term. Fields with zero coefficients may be left blank.

Blank Card

Run termination.

Deck Setup

The deck setup is shown in Figure 32.

Sample

A listing of the input cards for the sample problem is presented at the end of the sample main program.

Output

Description

The identification code, the polynomial order, the input coefficients, and the real and complex roots are printed.

Sample

The output listing of the sample problem is shown in Figure 33.

Program Modification

The maximum order of the polynomial acceptable to the sample program is fixed by the subroutine POLRT. However, input data in a different format can be handled by providing a specific format statement.

1. The sample program can accept polynomials up to the maximum 36th order, which is allowed by the subroutine.
2. Changes to the format of the parameter card and data cards can be made by modifying FORMAT statements 10 and 40, respectively, in main sample program SMPRT.

Operating Instructions

The polynomial roots sample program is a standard FORTRAN program. Special operating instructions are not required. Logical unit 2 is used for input, and logical unit 1 is used for output.

Error Messages

The following conditions will result in error messages:

1. The order of the polynomial specified in the control card is less than one: ORDER OF POLYNOMIAL LESS THAN ONE.

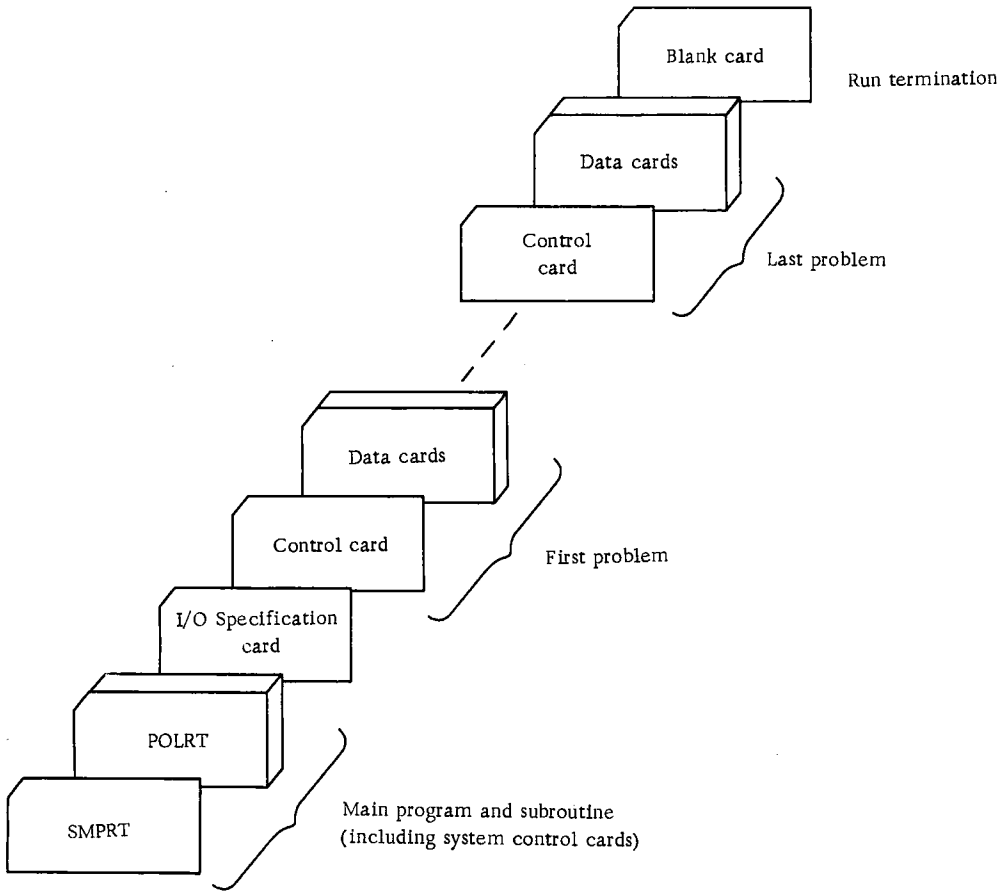


Figure 32. Deck setup (polynomial roots)

The program will go on to the next set of data.

2. The order of the polynomial specified in the control card is greater than 36: ORDER OF POLYNOMIAL GREATER THAN 36.

The program will go on to the next set of data.

```
REAL AND COMPLEX ROOTS OF A POLYNOMIAL USING SUBROUTINE POLRT

FOR POLYNOMIAL   360  OF ORDER   9
THE INPUT COEFFICIENTS ARE

 -0.10000000E 01   0.00000000E 00   0.00000000E 00   0.00000000E 00   0.00000000E 00   0.00000000E 00
  0.10000000E 01   0.00000000E 00   0.00000000E 00   0.10000000E 01


    REAL ROOT       COMPLEX ROOT

  0.29864880E 00   0.10045286E 01
  0.29864880E 00  -0.10045286E 01
 -0.10192700E 01   0.24362726E 00
 -0.10192700E 01  -0.24362726E 00
  0.91052586E 00   0.00000000E 00
  0.72062276E 00  -0.76090076E 00
  0.72062276E 00   0.76090076E 00
 -0.45526296E 00  -0.78853846E 00
 -0.45526296E 00   0.78853846E 00
```

Figure 33. Output listing

3. The subroutine POLRT is unable to determine a root after 500 iterations on eight different starting values: UNABLE TO DETERMINE ROOT. THOSE ALREADY FOUND ARE ...

The program will print all the roots that were computed and then go to the next set of data.

Sample Program for Real and Complex Roots of a Real Polynomial - SMPRT

Purpose:

Computes the real and complex roots of a real polynomial whose coefficients are input.

Remarks:

The order of the polynomial must be greater than one and less than thirty-seven.
I/O logical units determined by MX and MY, respectively.

Subroutines and function subprograms required:

POLRT

Method:

Reads a control card containing the identification code and the order of the polynomial whose coefficients are contained on the following data cards. The coefficients are then read and the roots are computed.

More than one control card and corresponding data can be processed. Execution is terminated by a blank control card.

```
// FOR
*IOCS(CARD,TYPEWRITER,1132 PRINTER)
*ONE WORD INTEGERS
C        SAMPLE PROGRAM FOR REAL AND COMPLEX ROOTS OF A REAL POLY-      SMPRT  1
C        NOMIAL - SMPRT                                                 SMPRT  2
      DIMENSION A(37),W(37),ROOTR(37),ROOTI(37)                         SMPRT  3
   10 FORMAT(1X,I4,3X,I2)                                               SMPRT  4
   30 FORMAT(////62H REAL AND COMPLEX ROOTS OF A POLYNOMIAL USING SUBROUSMPRT  5
     1TINE POLRT///17H FOR POLYNOMIAL  ,I4,2X,10HOF ORDER  ,I2//27H THE SMPRT  6
     2INPUT COEFFICIENTS ARE//)                                         SMPRT  7
   40 FORMAT(7F10.0)                                                    SMPRT  8
   50 FORMAT(6E16.7)                                                    SMPRT  9
   65 FORMAT(////34H ORDER OF POLYNOMIAL LESS THAN ONE)                 SMPRT 10
   77 FORMAT(////36H ORDER OF POLYNOMIAL GREATER THAN 36)               SMPRT 11
   79 FORMAT(////31H HIGH ORDER COEFFICIENT IS ZERO)                    SMPRT 12
   85 FORMAT(////50H UNABLE TO DETERMINE ROOT. THOSE ALREADY FOUND ARE) SMPRT 13
   95 FORMAT(////5X,9HREAL ROOT,6X,12HCOMPLEX ROOT//)                   SMPRT 14
   97 FORMAT(2E16.7)                                                    SMPRT 15
   98 FORMAT(2I2)                                                       SMPRT 16
      READ(2,98)MX,MY                                                   SMPRT 17
    5 READ(MY,10)ID,IORD                                                SMPRT 18
      IF(ID-IORD)100,100,20                                             SMPRT 19
   20 WRITE(MX,30)ID,IORD                                               SMPRT 20
      J=IORD+1                                                          SMPRT 21
      READ(MY,40)(A(I),I=1,J)                                           SMPRT 22
      WRITE(MX,50)(A(I),I=1,J)                                          SMPRT 23
      CALL POLRT(A,W,IORD,ROOTR,ROOTI,IER)                              SMPRT 24
      IF(IER-1)90,60,70                                                 SMPRT 25
   60 WRITE(MX,65)                                                      SMPRT 26
      GO TO 5                                                           SMPRT 27
   70 IF(IER-3)75,80,78                                                 SMPRT 28
   75 WRITE(MX,77)                                                      SMPRT 29
      GO TO 5                                                           SMPRT 30
   78 WRITE(MX,79)                                                      SMPRT 31
      GO TO 5                                                           SMPRT 32
   80 WRITE(MX,85)                                                      SMPRT 33
   90 WRITE(MX,95)                                                      SMPRT 34
      DO 96 I=1,IORD                                                    SMPRT 35
   96 WRITE(MX,97)ROOTR(I),ROOTI(I)                                     SMPRT 36
      GO TO 5                                                           SMPRT 37
  100 STOP                                                              SMPRT 38
      END                                                               SMPRT 39
// DUP
*STORE      WS  UA  SMPRT
// XEQ SMPRT
```

```
1 2                                                                     1
 360    9                                                               2
     -1.0                                    1.0                        3
                    1.0                                                 4
                                                                        5
```

SOLUTION OF SIMULTANEOUS EQUATIONS

Problem Description

A solution is obtained for a set of simultaneous equations by the method of elimination using largest pivotal divisor. Both the input data and the solution values are printed. This procedure is repeated until all sets of input data have been processed.

Program

Description

The solution of simultaneous equations sample program consists of a main routine, SOLN, and four subroutines:

SIMQ, LOC — are from the Scientific Subroutine Package

MATIN, MXOUT — are sample subroutines for matrix input and output

Capacity

The sample program will solve for 40 equations. The general rules for program modifications are described later.

Input

I/O Specification Card

A control card with the following format must precede each matrix of coefficients:

Columns	Contents	For Sample Problem
1 - 2	Blank	
3 - 6	Up to four-digit identification code (numeric only)	1
7 - 10	Number of rows in matrix	10
11 - 14	Number of columns in matrix (same as number of rows)	10

Each matrix must be followed by a card with a 9-punch in column 1. This, in turn, is followed by the constant vector.

Data Cards

Data cards are assumed to have seven fields of ten columns each. The decimal point may appear anywhere in a field, or be omitted, but the number must be right-justified. The number in each field may be preceded by blanks. Equation coefficients must be punched by row. A row may continue from card to card. However, each new row must start in the first field of the next card. The vector of constants is punched in continuous data fields following the 9 card. Columns 71 to 80 of data cards may be used for identification, sequence numbering, etc.

A blank card after the last set of input data terminates the run.

Deck Setup

The deck setup is shown in Figure 34.

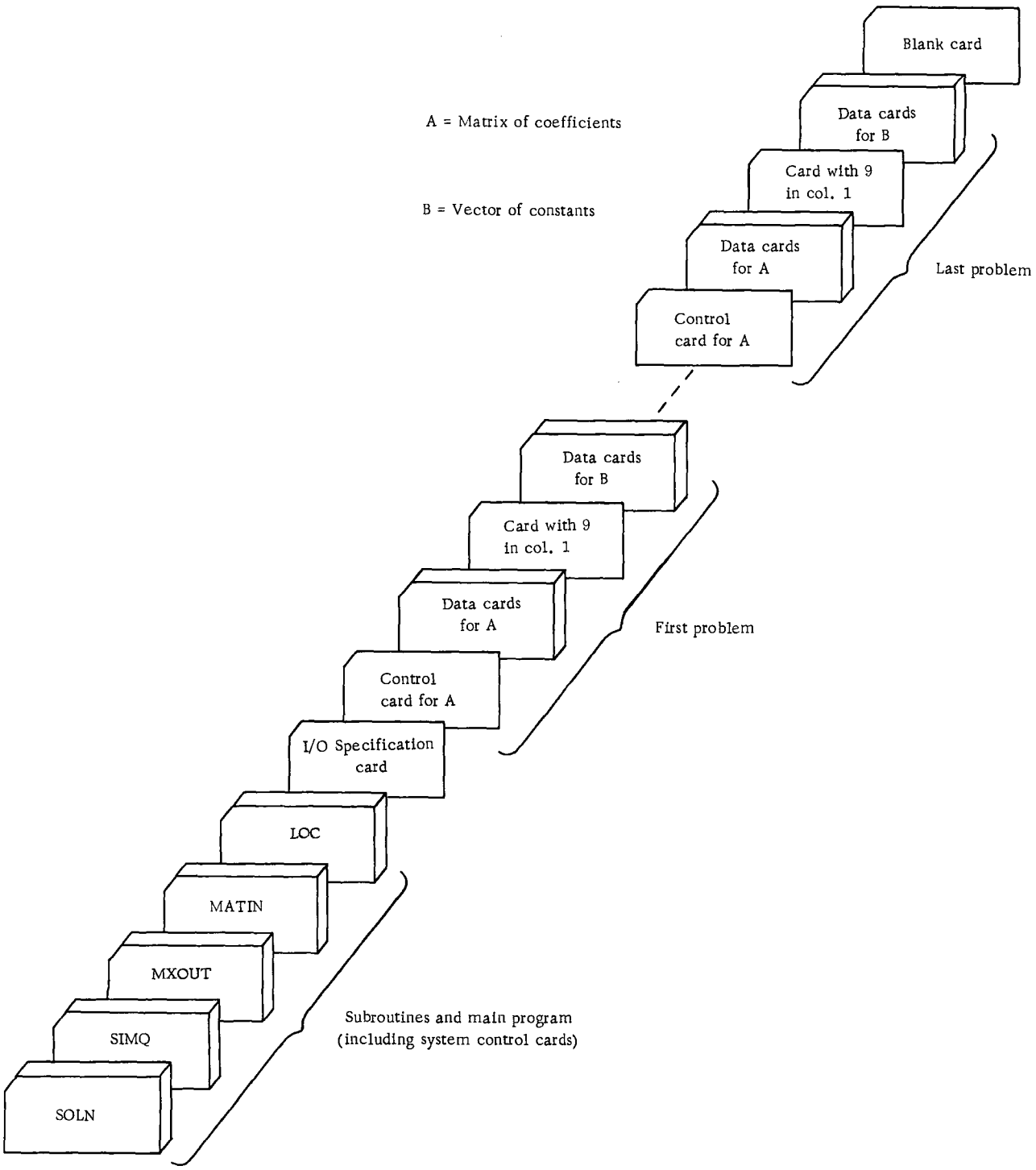


Figure 34. Deck setup (solution of simultaneous equations)

Sample

A listing of input cards for the sample problem is presented at the end of the sample main program.

Output

Description

The original matrix is printed for any sized array. Each six-column grouping is headed with the matrix code number, dimensions, and storage mode (always 0 in this sample program). Columns and rows are headed with their respective number. The original vector of constants is also printed. The solution values are then listed. This output is given for each case to be processed.

Sample

The output listing for the sample problem is shown in Figure 35.

Program Modifications

Noting that storage problems may result, as previously discussed in "Sample Program Description", the size of the maximum problem acceptable to the sample program can be increased. Output of the solution values in a different format can be handled by providing a specific format statement.

1. Changes to the DIMENSION statement of the main program, SOLN. The dimension of array A must be greater than or equal to the maximum number of elements in the matrix (N x N). The dimension of array B must be greater than or equal to N.
2. Insert the same number N in the third argument of the CALL MATIN statement (statement 25) in SOLN.
3. Changes to the format of the solution values may be made by modifying FORMAT statement 21 in SOLN.

```
SOLUTION OF SIMULTANEOUS EQUATIONS

 MATRIX    1     10 ROWS     10 COLUMNS     STORAGE MODE 0

          COLUMN      1             2             3             4             5             6
    ROW    1   0.100000E 01  0.664408E 00  0.760100E 00  0.750750E 00  0.429942E 00  0.332910E-02
    ROW    2   0.664408E 00  0.100000E 01  0.627180E 00  0.619465E 00  0.354757E 00  0.274700E-02
    ROW    3   0.760100E 00  0.627180E 00  0.100000E 01  0.708684E 00  0.405851E 00  0.314260E-02
    ROW    4   0.750750E 00  0.619465E 00  0.708684E 00  0.100000E 01  0.400859E 00  0.310390E-02
    ROW    5   0.429942E 00  0.354757E 00  0.405851E 00  0.400859E 00  0.100000E 01  0.177760E-02
    ROW    6   0.332910E-02  0.274700E-02  0.314260E-02  0.310390E-02  0.177760E-02  0.100000E 01
    ROW    7   0.728478E 00  0.601087E 00  0.687660E 00  0.679201E 00  0.388967E 00  0.301190E-02
    ROW    8   0.675176E 00  0.557106E 00  0.637344E 00  0.629504E 00  0.360507E 00  0.279150E-02
    ROW    9   0.863591E 00  0.712572E 00  0.815202E 00  0.805174E 00  0.461109E 00  0.357050E-02
    ROW   10   0.744684E 00  0.614459E 00  0.702958E 00  0.694310E 00  0.397620E 00  0.307890E-02

 MATRIX    1     10 ROWS     10 COLUMNS     STORAGE MODE 0

          COLUMN      7             8             9            10
    ROW    1   0.728478E 00  0.675176E 00  0.863591E 00  0.744684E 00
    ROW    2   0.601087E 00  0.557106E 00  0.712572E 00  0.614459E 00
    ROW    3   0.687660E 00  0.637344E 00  0.815202E 00  0.702958E 00
    ROW    4   0.679201E-00  0.629504E 00  0.805174E 00  0.694310E 00
    ROW    5   0.388967E 00  0.360507E 00  0.461109E 00  0.397620E 00
    ROW    6   0.301190E-02  0.279150E-02  0.357050E-02  0.307890E-02
    ROW    7   0.100000E 01  0.610829E 00  0.781287E 00  0.673713E 00
    ROW    8   0.610829E 00  0.100000E 01  0.724121E 00  0.624418E 00
    ROW    9   0.781287E 00  0.724121E 00  0.100000E 01  0.798668E 00
    ROW   10   0.673713E 00  0.624418E 00  0.798668E 00  0.100000E 01

ORIGINAL B VECTOR

     1          0.110000E 03
     2         -0.120000E 03
     3          0.100000E 02
     4          0.145000E 03
     5         -0.500000E 02
     6          0.442000E 02
     7         -0.140000E 02
     8          0.385000E 02
     9          0.220000E 02
    10          0.165000E 04

SOLUTION VALUES

     1         -0.283123E 03
     2         -0.567240E 03
     3         -0.516456E 03
     4         -0.299155E 02
     5         -0.179352E 03
     6          0.435178E 02
     7         -0.479274E 03
     8         -0.230431E 03
     9         -0.210172E 04
    10          0.480974E 04

END OF CASE
```

Figure 35. Output listing

The matrix listing is set for 120 print positions across the page, and double spacing. This can be changed by means of the last two arguments in the CALL MXOUT statement in SOLN (statement 65).

Operating Instructions

The sample program for the solution of simultaneous equations is a standard FORTRAN program. Special operating instructions are not required. Logical unit 2 is used for input, and logical unit 1 is used for output.

Error Messages

The following error conditions will result in messages:

1. Reserved storage area is too small for matrix: DIMENSIONED AREA TOO SMALL FOR

INPUT MATRIX (matrix code no.). GO ON TO NEXT CASE.

2. Matrix of coefficients is not square: ROW AND COLUMN DIMENSIONS NOT EQUAL FOR MATRIX (matrix code no.). GO ON TO NEXT CASE.

3. Number of data cards does not correspond to that required by parameter card: INCORRECT NUMBER OF DATA CARDS FOR MATRIX (matrix code no.). EXECUTION TERMINATED.

4. Singular input matrix: MATRIX IS SINGULAR. GO ON TO NEXT CASE.

Error conditions 1, 2, and 4 allow the computer run to continue. Error condition 3, however, terminates execution and requires another run to process succeeding cases.

Sample Main Program - SOLN

Purpose:

Solution of a set of simultaneous equations.

Remarks:

I/O specifications transmitted to subroutines by COMMON.

Input card:

Column 2 MX - Logical unit number for output.

Column 4 MY - Logical unit number for input.

Subroutines and function subprograms required:

SIMQ
MATIN
MXOUT
LOC

Method:

A matrix of simultaneous equations coefficients and a vector of constants are read from the standard input device. The solution is obtained and listed on the standard output device. This procedure is repeated for other sets of equations until a blank card is encountered.

```
// FOR
*IOCS(CARD,TYPEWRITER,1132 PRINTER)
*ONE WORD INTEGERS
C        SAMPLE MAIN PROGRAM - SOLN                                    SOLN   1
C        MATRIX IS DIMENSIONED FOR 1600 ELEMENTS. THEREFORE, NUMBER OF SOLN   2
C        EQUATIONS TO BE SOLVED CANNOT EXCEED 40 UNLESS DIMENSION      SOLN   3
C        STATEMENT IS CHANGED                                          SOLN   4
      DIMENSION A(1600),B(40)                                          SOLN   5
      COMMON MX,MY                                                     SOLN   6
   10 FORMAT(////35H SOLUTION OF SIMULTANEOUS EQUATIONS)               SOLN   7
   11 FORMAT(//45H DIMENSIONED AREA TOO SMALL FOR INPUT MATRIX ,I4)    SOLN   8
   12 FORMAT(//21H EXECUTION TERMINATED)                              SOLN   9
   13 FORMAT(//48H ROW AND COLUMN DIMENSIONS NOT EQUAL FOR MATRIX ,I4) SOLN  10
   14 FORMAT(//43H INCORRECT NUMBER OF DATA CARDS FOR MATRIX ,I4)      SOLN  11
   15 FORMAT(//19H GO ON TO NEXT CASE)                                SOLN  12
   16 FORMAT(//39H STRUCTURE CODE IS NOT ZERO FOR MATRIX,I4)          SOLN  13
   17 FORMAT(/////18H ORIGINAL B VECTOR,/////)                        SOLN  14
   18 FORMAT(////16H SOLUTION VALUES,/////)                           SOLN  15
   19 FORMAT(//19H MATRIX IS SINGULAR)                                SOLN  16
   20 FORMAT(7F10.0)                                                  SOLN  17
   21 FORMAT(I6,10X,E16.6)                                            SOLN  18
   22 FORMAT(//12H END OF CASE)                                       SOLN  19
   23 FORMAT(2I2)                                                     SOLN  20
      READ(2,23)MX,MY                                                 SOLN  21
      WRITE(MX,10)                                                    SOLN  22
   25 CALL MATIN (ICOD,A,1600,N,M,MS,IER)                             SOLN  23
      IF(N) 30,95,30                                                  SOLN  24
   30 IF(IER-1) 45,35,40                                              SOLN  25
   35 WRITE(MX,11)ICOD                                                SOLN  26
      GO TO 90                                                        SOLN  27
   40 WRITE(MX,14)ICOD                                                SOLN  28
      GO TO 95                                                        SOLN  29
   45 IF(N-M) 50,55,50                                                SOLN  30
   50 WRITE(MX,13)ICOD                                                SOLN  31
      GO TO 90                                                        SOLN  32
   55 IF(MS) 60,65,60                                                 SOLN  33
   60 WRITE(MX,16)ICOD                                                SOLN  34
      GO TO 90                                                        SOLN  35
   65 CALL MXOUT(ICOD,A,N,M,MS,60,120,2)                              SOLN  36
      READ(MY,20)(B(I),I=1,N)                                         SOLN  37
      WRITE(MX,17)                                                    SOLN  38
      DO 70 I=1,N                                                     SOLN  39
   70 WRITE(MX,21)I,B(I)                                              SOLN  40
      CALL SIMQ(A,B,N,KS)                                             SOLN  41
      IF(KS-1) 80,75,80                                               SOLN  42
   75 WRITE(MX,19)                                                    SOLN  43
      WRITE(MX,15)                                                    SOLN  44
      GO TO 25                                                        SOLN  45
   80 WRITE(MX,18)                                                    SOLN  46
      DO 85 I=1,N                                                     SOLN  47
   85 WRITE(MX,21)I,B(I)                                              SOLN  48
      WRITE(MX,22)                                                    SOLN  49
      GO TO 25                                                        SOLN  50
   90 READ(MY,20)(B(I),I=1,N)                                         SOLN  51
      WRITE(MX,15)                                                    SOLN  52
      GO TO 25                                                        SOLN  53
   95 WRITE(MX,12)                                                    SOLN  54
      STOP                                                            SOLN  55
      END                                                             SOLN  56
// DUP
*STORE      WS  UA  SOLN
// XEQ SOLN
```

```
1 2                                                                              1
 0001001000010                                                                   2
1.0000000 0.6644085 0.7601008 0.7507505 0.4299425 0.0033291 0.7284786           3
0.6751766 0.8635910 0.7446845                                                   4
0.6644085 1.0000000 0.6271802 0.6194650 0.3547574 0.0027470 0.6010878           5
0.5571068 0.7125728 0.6144597                                                   6
0.7601008 0.6271802 1.0000000 0.7086843 0.4058519 0.0031426 0.6876602           7
0.6373449 0.8152021 0.7029582                                                   8
0.7507505 0.6194650 0.7086843 1.0000000 0.4008593 0.0031039 0.6792011           9
0.6295047 0.8051740 0.6943108                                                  10
0.4299425 0.3547574 0.4058519 0.4008593 1.0000000 0.0017776 0.3889673          11
0.3605070 0.4611099 0.3976204                                                  12
0.0033291 0.0027470 0.0031426 0.0031039 0.0017776 1.0000000 0.0030119          13
0.0027915 0.0035705 0.0030789                                                  14
0.7284786 0.6010878 0.6876602 0.6792011 0.3889673 0.0030119 1.0000000          15
0.6108296 0.7812874 0.6737132                                                  16
0.6751766 0.5571068 0.6373449 0.6295047 0.3605070 0.0027915 0.6108296          17
1.0000000 0.7241215 0.6244183                                                  18
0.8635910 0.7125728 0.8152021 0.8051740 0.4611099 0.0035705 0.7812874          19
0.7241215 1.0000000 0.7986682                                                  20
0.7446845 0.6144597 0.7029582 0.6943108 0.3976204 0.0030789 0.6737132          21
0.6244183 0.7986682 1.0000000                                                  22
9                                                                              23
     110.0    -120.0      10.      145.      -50.      44.20      -14.         24
      38.5      22.     1650.                                                  25
                                                                               26
```

```
SUBROUTINE MATIN

PURPOSE
   READS CONTROL CARD AND MATRIX DATA ELEMENTS FROM LOGICAL
   UNIT 5

USAGE
   CALL MATIN(ICODE,A,ISIZE,IROW,ICOL,IS,IER)

DESCRIPTION OF PARAMETERS
   ICODE-UPON RETURN, ICODE WILL CONTAIN FOUR DIGIT
         IDENTIFICATION CODE FROM MATRIX PARAMETER CARD
   A     -DATA AREA FOR INPUT MATRIX
   ISIZE-NUMBER OF ELEMENTS DIMENSIONED BY USER FOR AREA A
   IROW -UPON RETURN, IROW WILL CONTAIN ROW DIMENSION FROM
         MATRIX PARAMETER CARD
   ICOL -UPON RETURN, ICOL WILL CONTAIN COLUMN DIMENSION FROM
         MATRIX PARAMETER CARD
   IS   -UPON RETURN, IS WILL CONTAIN STORAGE MODE CODE FROM
         MATRIX PARAMETER CARD WHERE
         IS=0 GENERAL MATRIX
         IS=1 SYMMETRIC MATRIX
         IS=2 DIAGONAL MATRIX
   IER  -UPON RETURN, IER WILL CONTAIN AN ERROR CODE WHERE
         IER=0   NO ERROR
         IER=1   ISIZE IS LESS THAN NUMBER OF ELEMENTS IN
                 INPUT MATRIX
         IER=2   INCORRECT NUMBER OF DATA CARDS

REMARKS
   NONE

SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
   LOC

METHOD
   SUBROUTINE ASSUMES THAT INPUT MATRIX CONSISTS OF PARAMETER
   CARD FOLLOWED BY DATA CARDS
   PARAMETER CARD HAS THE FOLLOWING FORMAT
     COL. 1- 2 BLANK
     COL. 3- 6 UP TO FOUR DIGIT IDENTIFICATION CODE
     COL. 7-10 NUMBER OF ROWS IN MATRIX
     COL.11-14 NUMBER OF COLUMNS IN MATRIX
     COL.15-16 STORAGE MODE OF MATRIX WHERE
          0 - GENERAL MATRIX
          1 - SYMMETRIC MATRIX
          2 - DIAGONAL MATRIX
   DATA CARDS ARE ASSUMED TO HAVE SEVEN FIELDS OF TEN COLUMNS
   EACH.  DECIMAL POINT MAY APPEAR ANYWHERE IN A FIELD.  IF NO
   DECIMAL POINT IS INCLUDED, IT IS ASSUMED THAT THE DECIMAL
   POINT IS AT THE END OF THE 10 COLUMN FIELD. NUMBER IN EACH
   FIELD MAY BE PRECEDED BY BLANKS.  DATA ELEMENTS MUST BE
   PUNCHED BY ROW.  A ROW MAY CONTINUE FROM CARD TO CARD.
   HOWEVER EACH NEW ROW MUST START IN THE FIRST FIELD OF THE
   NEXT CARD.  ONLY THE UPPER TRIANGULAR PORTION OF A SYMMETRIC
   OR THE DIAGONAL ELEMENTS OF A DIAGONAL MATRIX ARE CONTAINED
   ON DATA CARDS.  THE FIRST ELEMENT OF EACH NEW ROW WILL BE
   THE DIAGONAL ELEMENT FOR A MATRIX WITH  SYMMETRIC OR
   DIAGONAL STORAGE MODE. COLUMNS 71-80 OF DATA CARDS MAY BE
   USED FOR IDENTIFICATION, SEQUENCE NUMBERING, ETC..
   THE LAST DATA CARD FOR ANY MATRIX MUST BE FOLLOWED BY A CARD
   WITH A 9 PUNCH IN COLUMN 1.
```

```
      SUBROUTINE MATIN(ICODE,   A,ISIZE,IROW,ICOL,IS,IER)                MATIN  1
      DIMENSION A(1)                                                     MATIN  2
      DIMENSION CARD(8)                                                  MATIN  3
      COMMON MX,MY                                                       MATIN  4
    1 FORMAT(7F10.0)                                                     MATIN  5
    2 FORMAT(I6,2I4,I2)                                                  MATIN  6
    3 FORMAT(I1)                                                         MATIN  7
      IDC=7                                                              MATIN  8
      IER=0                                                              MATIN  9
      READ( MY,2)ICODE,IROW,ICOL,IS                                      MATIN 10
      CALL LOC(IROW,ICOL,ICNT,IROW,ICOL,IS)                              MATIN 11
      IF(ISIZE-ICNT)6,7,7                                                MATIN 12
    6 IER=1                                                              MATIN 13
    7 IF (ICNT)38,38,8                                                   MATIN 14
    8 ICOLT=ICOL                                                         MATIN 15
      IROCR=1                                                            MATIN 16
C        COMPUTE NUMBER OF CARDS FOR THIS ROW                            MATIN 17
   11 IRCDS=(ICOLT-1)/IDC+1                                              MATIN 18
      IF(IS-11)15,15,12                                                  MATIN 19
   12 IRCDS=1                                                            MATIN 20
C        SET UP LOOP FOR NUMBER OF CARDS IN ROW                          MATIN 21
   15 DO 31 K=1,IRCDS                                                    MATIN 22
      READ(MY,1)(CARD(I),I=1,IDC)                                        MATIN 23
C        SKIP THROUGH DATA CARDS IF INPUT AREA TOO SMALL                 MATIN 24
      IF(IER)16,16,31                                                    MATIN 25
   16 L=0                                                                MATIN 26
C        COMPUTE COLUMN NUMBER FOR FIRST FIELD IN CURRENT CARD           MATIN 27
      JS=(K-1)*IDC+ICOL-ICOLT+1                                          MATIN 28
```

```
      SUBROUTINE MXOUT (ICODE,A,N,M,MS,LINS,IPOS,ISP)                    MXOUT  1
      DIMENSION A(1),B(8)                                                MXOUT  2
      COMMON MX,MY                                                       MXOUT  3
    1 FORMAT(////5X, 7HMATRIX ,I5,6X,I3,5H ROWS,6X,I3,8H COLUMNS,        MXOUT  4
     18X,13HSTORAGE MODE ,I1,/)                                          MXOUT  5
    2 FORMAT(12X,8HCOLUMN  ,7(3X,I3,10X)//)                             MXOUT  6
    4 FORMAT(7X,4HROW ,I3,7(E16.6))                                      MXOUT  7
    5 FORMAT(/,7X,4HROW ,I3,7(E16.6))                                    MXOUT  8
      J=1                                                                MXOUT  9
C        WRITE HEADING                                                   MXOUT 10
      NEND=IPOS/16-1                                                     MXOUT 11
      LEND = (LINS/ISP)-10                                               MXOUT 12
   10 LSTRT=1                                                            MXOUT 13
   20 WRITE(MX,1)ICODE,N,M,MS                                            MXOUT 14
      JNT=J+NEND-1                                                       MXOUT 15
      IF(JNT-M)33,33,32                                                  MXOUT 16
   32 JNT=M                                                              MXOUT 17
   33 CONTINUE                                                           MXOUT 18
      WRITE(MX,2)(JCUR,JCUR=J,JNT)                                       MXOUT 19
      LTEND = LSTRT+LEND-1                                               MXOUT 20
      DO 80 L=LSTRT,LTEND                                                MXOUT 21
C        FORM OUTPUT ROW LINE                                            MXOUT 22
      DO 55 K=1,NEND                                                     MXOUT 23
      KK=K                                                               MXOUT 24
      JT = J+K-1                                                         MXOUT 25
      CALL LOC(L,JT,IJNT,N,M,MS)                                         MXOUT 26
      B(K)=0.0                                                           MXOUT 27
      IF(IJNT)50,50,45                                                   MXOUT 28
   45 B(K)=A(IJNT)                                                       MXOUT 29
   50 CONTINUE                                                           MXOUT 30
C        CHECK IF LAST COLUMN.  IF YES GO TO 60                          MXOUT 31
      IF(JT-M) 55,60,60                                                  MXOUT 32
   55 CONTINUE                                                           MXOUT 33
C        END OF LINE, NOW WRITE                                          MXOUT 34
   60 IF(ISP-1)65,65,70                                                  MXOUT 35
   65 WRITE(MX,41L,(B(JW),JW=1,KK)                                       MXOUT 36
      GO TO 75                                                           MXOUT 37
   70 WRITE(MX,5)L,(B(JW),JW=1,KK)                                       MXOUT 38
C        IF END OF ROWS,GO CHECK COLUMNS                                 MXOUT 39
   75 IF(N-L)85,85,80                                                    MXOUT 40
   80 CONTINUE                                                           MXOUT 41
C        WRITE NEW HEADING                                               MXOUT 42
      LSTRT=LSTRT+LEND                                                   MXOUT 43
      GO TO 20                                                           MXOUT 44
C        END OF COLUMNS, THEN RETURN                                     MXOUT 45
   85 IF(JT-M)90,95,95                                                   MXOUT 46
   90 J=JT+1                                                             MXOUT 47
      GO TO 10                                                           MXOUT 48
   95 RETURN                                                             MXOUT 49
      END                                                                MXOUT 50
```

USAGE

CALL MXOUT(ICODE,A,N,M,MS,LINS,IPOS,ISP)

DESCRIPTION OF PARAMETERS

ICODE- INPUT CODE NUMBER TO BE PRINTED ON EACH OUTPUT PAGE
A-NAME OF OUTPUT MATRIX
N-NUMBER OF ROWS IN A
M-NUMBER OF COLUMNS IN A
MS-STORAGE MODE OF A WHERE MS=
0-GENERAL
1-SYMMETRIC
2-DIAGONAL
LINS-NUMBER OF PRINT LINES ON THE PAGE (USUALLY 60)
IPOS-NUMBER OF PRINT POSITIONS ACROSS THE PAGE (USUALLY 132)
ISP-LINE SPACING CODE, 1 FOR SINGLE SPACE, 2 FOR DOUBLE SPACE

REMARKS

NONE

SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED

LOC

METHOD

THIS SUBROUTINE CREATES A STANDARD OUTPUT LISTING OF ANY SIZED ARRAY WITH ANY STORAGE MODE. EACH PAGE IS HEADED WITH THE CODE NUMBER,DIMENSIONS AND STORAGE MODE OF THE ARRAY. EACH COLUMN AND ROW IS ALSO HEADED WITH ITS RESPECTIVE NUMBER.

SUBROUTINE MXOUT

PURPOSE

PRODUCES AN OUTPUT LISTING OF ANY SIZED ARRAY ON LOGICAL UNIT 1

Printed in U.S.A. H20-0252-3

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)