



## **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

© Copyright International Business Machines Corporation 1967, 1968

## CONTENTS

|                                 |   |                                           |     |
|---------------------------------|---|-------------------------------------------|-----|
| Introduction .....              | 1 | Program Modification .....                | 7   |
| Areas of Application .....      | 1 | Optimization of Time .....                | 7   |
| Statistics .....                | 1 | Extended Precision .....                  | 7   |
| Matrix Manipulation .....       | 1 | Format of the Documentation .....         | 8   |
| Other Mathematical Areas .....  | 1 | Subroutine Descriptions .....             | 8   |
| Characteristics .....           | 1 | Sample Program Descriptions .....         | 8   |
| Design Philosophy .....         | 2 | Machine Configuration .....               | 8   |
| Choice of Algorithms .....      | 2 | Guide to Subroutines .....                | 9   |
| Programming .....               | 2 | Subroutine Listings and Writeups .....    | 13  |
| Overall Rules of Usage .....    | 3 |                                           |     |
| General Rules .....             | 3 | Appendix A. Storage Requirements .....    | 128 |
| Matrix Operations .....         | 3 | Appendix B. Accuracy of Subroutines ..... | 133 |
| Variable Dimensioning .....     | 3 | Appendix C. Timing .....                  | 137 |
| Storage Compression .....       | 4 | Appendix D. Sample Programs .....         | 138 |
| Matrix Element References ..... | 5 |                                           |     |



## 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 inter-correlations 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.

|     |   | Column |      |      |      |      |   |   |   |   |    |
|-----|---|--------|------|------|------|------|---|---|---|---|----|
|     |   | 1      | 2    | 3    | 4    | 5    | 6 | 7 | 8 | 9 | 10 |
| Row | 1 | (1)    | (6)  | (11) | (16) | (21) |   |   |   |   |    |
|     | 2 | (2)    | (7)  | (12) | (17) | (22) |   |   |   |   |    |
|     | 3 | (3)    | (8)  | (13) | (18) | (23) |   |   |   |   |    |
|     | 4 | (4)    | (9)  | (14) | (19) | (24) |   |   |   |   |    |
|     | 5 | (5)    | (10) | (15) | (20) | (25) |   |   |   |   |    |
| 6   |   |        |      |      |      |      |   |   |   |   |    |
| 7   |   |        |      |      |      |      |   |   |   |   |    |
| 8   |   |        |      |      |      |      |   |   |   |   |    |
| 9   |   |        |      |      |      |      |   |   |   |   |    |
| 10  |   |        |      |      |      |      |   |   |   |   |    |

Figure 1. Double subscripted data storage

|      |      |      |
|------|------|------|
| (1)  | (11) | (21) |
| (2)  | (12) | (22) |
| (3)  | (13) | (23) |
| (4)  | (14) | (24) |
| (5)  | (15) | (25) |
| (6)  | (16) |      |
| (7)  | (17) |      |
| (8)  | (18) |      |
| (9)  | (19) |      |
| (10) | (20) |      |

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.

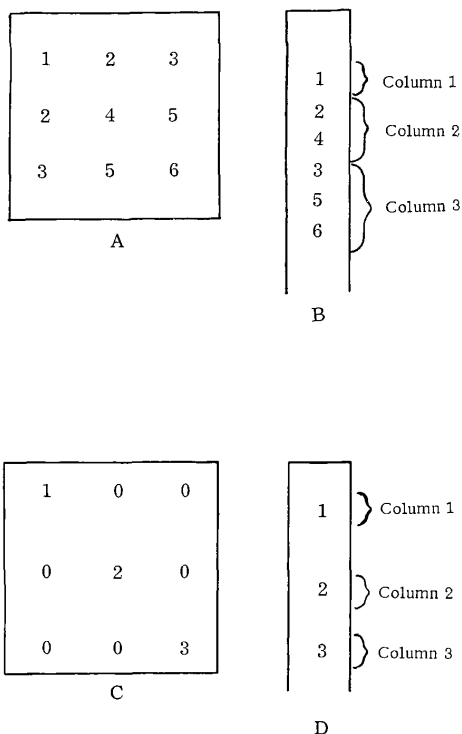


Figure 3. Storage mode

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 -----
```

Overall Rules of Usage 5

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, LJ, 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, LJ, 3, 3, 1)
```

will result in LJ 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.

## PROGRAM MODIFICATION

### 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 MACX(A,B,P,N,M,MSA,MSB)
  DIMENSION A(1),B(1),R(1)
  C
  C      TEST FOR SAME STORAGE MODE
  C
  IF(MSA-MSB) 30,1C,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 GO 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) 70,75,9C
  70 IR=KX*(J-1)+1
  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(IRX) 40,45,50
  ENC
```

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

|                                                                      | <u>Page</u> |                                                   | <u>Page</u> |
|----------------------------------------------------------------------|-------------|---------------------------------------------------|-------------|
| <u>STATISTICS</u>                                                    |             | <u>Analysis of Variance</u>                       |             |
| <u>Data Screening</u>                                                |             | AVDAT--data storage allocation                    | 34          |
| TALLY--totals, means, standard deviations, minimums, and maximums    | 13          | AVCAL-- $\Sigma$ and $\Delta$ operation           | 35          |
| BOUND--selection of observations within bounds                       | 14          | MEANQ--mean square operation                      | 36          |
| SUBST--subset selection from observation matrix                      | 15          | <u>Discriminant Analysis</u>                      |             |
| ABSNT--detection of missing data                                     | 16          | DMATX--means and dispersion matrix                | 38          |
| TAB1--tabulation of data (1 variable)                                | 16          | DISCR--discriminant functions                     | 39          |
| TAB2--tabulation of data (2 variables)                               | 18          | <u>Factor Analysis</u>                            |             |
| SUBMX--build subset matrix                                           | 20          | TRACE--cumulative percentage of eigenvalues       | 41          |
| <u>Elementary Statistics</u>                                         |             | LOAD--factor loading                              | 42          |
| MOMEN--first four moments                                            | 20          | VARMX--varimax rotation                           | 43          |
| TTSTT--tests on population means                                     | 21          | <u>Time Series</u>                                |             |
| <u>Correlation</u>                                                   |             | AUTO--autocovariances                             | 46          |
| CORRE--means, standard deviations, and correlations                  | 23          | CROSS--crosscovariances                           | 47          |
| <u>Multiple Linear Regression</u>                                    |             | SMO--application of filter coefficients (weights) | 48          |
| ORDER--rearrangement of inter-correlations                           | 25          | EXSMO--triple exponential smoothing               | 49          |
| MULTR--multiple regression and correlation                           | 26          | <u>Nonparametric Statistics</u>                   |             |
| <u>Polynomial Regression</u>                                         |             | CHISQ-- $\chi^2$ test for a contingency table     | 50          |
| GDATA--data generation                                               | 28          | UTEST--Mann-Whitney U-test                        | 52          |
| <u>Canonical Correlation</u>                                         |             | TWOAV--Friedman two-way analysis of variance      | 53          |
| CANOR--canonical correlation                                         | 30          | QTEST--Cochran Q-test                             | 54          |
| NROOT--eigenvalues and eigenvectors of a special nonsymmetric matrix | 32          | SRANK--Spearman rank correlation                  | 55          |
|                                                                      |             | KRANK--Kendall rank correlation                   | 56          |
|                                                                      |             | WTEST--Kendall coefficient of concordance         | 58          |

|                                                 |             |                                 |             |
|-------------------------------------------------|-------------|---------------------------------|-------------|
|                                                 | <u>Page</u> |                                 | <u>Page</u> |
| RANK--rank observations                         | 59          | <u>Random Number Generators</u> |             |
| TIE--calculation of ties in ranked observations | 59          | RANDU--uniform random numbers   | 60          |
|                                                 |             | GAUSS--normal random numbers    | 60          |

|                                                                    | <u>Page</u> |                                                                                                 | <u>Page</u> |
|--------------------------------------------------------------------|-------------|-------------------------------------------------------------------------------------------------|-------------|
| <u>MATHEMATICS OPERATIONS</u>                                      |             | CINT--interchange two columns                                                                   | 74          |
| <u>Special Matrix Operations</u>                                   |             | RSUM--sum the rows of a matrix                                                                  | 74          |
| MINV--matrix inversion                                             | 61          | CSUM--sum the columns of a matrix                                                               | 75          |
| EIGEN--eigenvalues and eigenvectors of a<br>real, symmetric matrix | 62          | RTAB--tabulate the rows of a matrix                                                             | 75          |
| <u>Matrices</u>                                                    |             | CTAB--tabulate the columns of a matrix                                                          | 76          |
| GMADD--add two general matrices                                    | 64          | RSRT--sort matrix rows                                                                          | 77          |
| GMSUB--subtract two general matrices                               | 64          | CSRT--sort matrix columns                                                                       | 78          |
| GMPRD--product of two general matrices                             | 65          | RCUT--partition row-wise                                                                        | 79          |
| GMTRA--transpose of a general matrix                               | 65          | CCUT--partition column-wise                                                                     | 79          |
| GTPRD--transpose product of two general<br>matrices                | 66          | RTIE--adjoin two matrices row-wise                                                              | 80          |
| MADD--add two matrices                                             | 66          | CTIE--adjoin two matrices column-wise                                                           | 80          |
| MSUB--subtract two matrices                                        | 67          | MCPY--matrix copy                                                                               | 81          |
| MPRD--matrix product (row into column)                             | 67          | XCPY--copy submatrix from given matrix                                                          | 81          |
| MTRA--transpose a matrix                                           | 68          | RCPY--copy row of matrix into vector                                                            | 82          |
| TPRD--transpose product                                            | 68          | CCPY--copy column of matrix into vector                                                         | 82          |
| MATA--transpose product of matrix by<br>itself                     | 69          | DCPY--copy diagonal of matrix into vector                                                       | 83          |
| SADD--add scalar to matrix                                         | 69          | SCLA--matrix clear and add scalar                                                               | 83          |
| SSUB--subtract scalar from a matrix                                | 70          | DCLA--replace diagonal with scalar                                                              | 84          |
| SMPY--matrix multiplied by a scalar                                | 70          | MSTR--storage conversion                                                                        | 84          |
| SDIV--matrix divided by a scalar                                   | 71          | MFUN--matrix transformation by a<br>function                                                    | 85          |
| RADD--add row of one matrix to row of<br>another matrix            | 71          | RECP--reciprocal function for MFUN                                                              | 85          |
| CADD--add column of one matrix to col-<br>umn of another matrix    | 72          | LOC--location in compressed-stored<br>matrix                                                    | 86          |
| SRMA--scalar multiply row and add to<br>another row                | 72          | ARRAY--vector storage-double dimen-<br>sioned storage conversion                                | 86          |
| SCMA--scalar multiply column and add to<br>another column          | 73          | <u>Integration and Differentiation</u>                                                          |             |
| RINT--interchange two rows                                         | 73          | QSF--integral of equidistantly tabulated<br>function by Simpson's Rule                          | 87          |
|                                                                    |             | QATR--integral of given function by<br>trapezoidal rule using Romberg's<br>extrapolation method | 88          |

|                                                                                                                      | <u>Page</u> |                                                                      | <u>Page</u> |
|----------------------------------------------------------------------------------------------------------------------|-------------|----------------------------------------------------------------------|-------------|
| <u>Ordinary Differential Equations</u>                                                                               |             | <u>Nonlinear Equations</u>                                           |             |
| RK1--integral of first-order differential equation by Runge-Kutta method                                             | 90          | RTWI--refine estimate of root by Wegstein's iteration                | 116         |
| RK2--tabulated integral of first-order differential equation by Runge-Kutta method                                   | 91          | RTMI--determine root within a range by Mueller's iteration           | 117         |
| RKGS--solution of a system of first-order differential equations with given initial values by the Runge-Kutta method | 92          | RTNI--refine estimate of root by Newton's iteration                  | 119         |
| <u>Fourier Analysis</u>                                                                                              |             | <u>Roots of Polynomial</u>                                           |             |
| FORIF--Fourier analysis of a given function                                                                          | 95          | POLRT--real and complex roots of a real polynomial                   | 120         |
| FORIT--Fourier analysis of a tabulated function                                                                      | 96          | <u>Polynomial Operations</u>                                         |             |
| <u>Special Operations and Functions</u>                                                                              |             | PADD--add two polynomials                                            | 122         |
| GAMMA--gamma function                                                                                                | 97          | PADDM--multiply polynomial by constant and add to another polynomial | 122         |
| LEP--Legendre polynomial                                                                                             | 98          | PCLA--replace one polynomial by another                              | 122         |
| BESJ--J Bessel function                                                                                              | 99          | PSUB--subtract one polynomial from another                           | 123         |
| BESY--Y Bessel function                                                                                              | 101         | PMPY--multiply two polynomials                                       | 123         |
| BESI--I Bessel function                                                                                              | 103         | PDIV--divide one polynomial by another                               | 124         |
| BESK--K Bessel function                                                                                              | 104         | PQSD--quadratic synthetic division of a polynomial                   | 124         |
| CEL1--elliptic integral of the first kind                                                                            | 105         | PVAL--value of a polynomial                                          | 124         |
| CEL2--elliptic integral of the second kind                                                                           | 106         | PVSUB--substitute variable of polynomial by another polynomial       | 125         |
| EXPI--exponential integral                                                                                           | 108         | PCLD--complete linear synthetic division                             | 125         |
| SICI--sine cosine integral                                                                                           | 110         | PILD--evaluate polynomial and its first derivative                   | 125         |
| CS--Fresnel integrals                                                                                                | 112         | PDER--derivative of a polynomial                                     | 126         |
| <u>Linear Equations</u>                                                                                              |             | PINT--integral of a polynomial                                       | 126         |
| SIMQ--solution of simultaneous linear, algebraic equations                                                           | 115         | PGCD--greatest common divisor of two polynomials                     | 126         |
|                                                                                                                      |             | PNORM--normalize coefficient vector of polynomial                    | 127         |

## SUBROUTINE LISTINGS AND WRITEUPS

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)
DIMENSION A(1),S(1),TOTAL(1),AVER(1),SD(1),VMIN(1),VMAX(1)
C CLEAR OUTPUT VECTORS AND INITIALIZE VMIN,VMAX
DO 1 K=1,NV
TOTAL(K)=0.0
AVER(K)=0.0
SD(K)=0.0
VMIN(K)=1.0E30
VMAX(K)=1.0E30
1 VMAX(K)=1.0E30
C TEST SUBSET VECTOR
SCNT=0.0
DO 2 J=1,NO
IF(S(J)) 2,7,2
IF(15(J)) 2,7,2
2 SCNT=SCNT+1.0
3 CALCULATE TOTAL, MINIMA, MAXIMA
DO 6 I=1,NV
I2=I+NO
TOTAL(I2)=TOTAL(I)+A(I2)
IF(A(I2)-VMIN(I)) 3,4,4
3 VMIN(I2)=A(I2)
4 IF(A(I2)-VMAX(I)) 6,6,5
5 VMAX(I2)=A(I2)
6 SD(I2)=SD(I)+A(I2)*A(I2)
7 CONTINUE
C CALCULATE MEANS AND STANDARD DEVIATIONS
DO 8 I=1,NV
AVER(I)=TOTAL(I)/SCNT
8 SD(I)=SQRT(ARS((SD(I)-TOTAL(I)*TOTAL(I)/SCNT)/(SCNT-1.0)))
RETURN
END
TALLY 1
TALLY 2
TALLY 3
TALLY 4
TALLY 5
TALLY 6
TALLY 7
TALLY 8
TALLY 9
TALLY 10
TALLY 11
TALLY 12
TALLY 13
TALLY 14
TALLY 15
TALLY 16
TALLY 17
TALLY 18
TALLY 19
TALLY 20
TALLY 21
TALLY 22
TALLY 23
TALLY 24
TALLY 25
TALLY 26
TALLY 27
TALLY 28
TALLY 29
TALLY 30
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)
DIMENSION A(1),S(1),BLO(1),BHI(1),UNDER(1),BETW(1),OVER(1)
C CLEAR OUTPUT VECTORS.
DO 1 K=1,NV
  UNDER(K)=0.0
  BETW(K)=0.0
  OVER(K)=0.0
C TEST SUBSET VECTOR
DO 2 J=1,NO
  IF(S(J)) 2,8,2
C COMPARE OBSERVATIONS WITH BOUNDS
DO 7 I=1,NV
  IF(A(I,J)-BLO(I)) 5,3,3
  IF(A(I,J)-BHI(I)) 4,4,6
C COUNT
  IF(BETW(I)=BETW(I)+1.0
  GO TO 7
  IF(UNDER(I)=UNDER(I)+1.0
  GO TO 7
  IF(OVER(I)=OVER(I)+1.0
  CONTINUE
  CONTINUE
  RETURN
END
```

ROUND 1  
ROUND 2  
ROUND 3  
ROUND 4  
ROUND 5  
ROUND 6  
ROUND 7  
ROUND 8  
ROUND 9  
ROUND 10  
ROUND 11  
ROUND 12  
ROUND 13  
ROUND 14  
ROUND 15  
ROUND 16  
ROUND 17  
ROUND 18  
ROUND 19  
ROUND 20  
ROUND 21  
ROUND 22  
ROUND 23  
ROUND 24  
ROUND 25  
ROUND 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)
DIMENSION A(1),C(1),R(1),S(1)
DO 9 I=1,NO
  IG0=1-NO
  K=2
  DO 8 J=1,NC
    CLEAR R VECTOR
    R(I)=0
    LOCATE ELEMENT IN OBSERVATION MATRIX AND RELATIONAL CODE
    K=K+3
    IF(C(1))
      IA=IG0+I2*NO
      IG0=C(K+1)
      FORM R VECTOR
      Q=A(IA)-C(K+2)
      GO TO(1,2,3,4,5,5),IG0
1 IF(Q) 7,8,8
2 IF(Q) 7,7,8
3 IF(Q) 8,7,8
4 IF(Q) 7,0,7
5 IF(Q) 8,7,7
6 IF(Q) 8,0,7
7 RIJ=1,0
8 CONTINUE
    CALCULATE S VECTOR
9 CALL R(R,S(1))
RETURN
END
```

```
SUBST 1
SUBST 2
SUBST 1
SUBST 4
SUBST 5
SUBST 6
SUBST 7
SUBST 8
SUBST 9
SUBST 10
SUBST 11
SUBST 12
SUBST 13
SUBST 14
SUBST 15
SUBST 16
SUBST 17
SUBST 18
SUBST 19
SUBST 20
SUBST 21
SUBST 22
SUBST 23
SUBST 24
SUBST 25
SUBST 26
SUBST 27
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)
  DIMENSION A(1),S(1)
  DO 20 J=1,NV
    IJ=J-NV
    S(J)=1
    DO 10 I=1,NO
      IJ=IJ+NO
      IF (A(IJ)) 10,5,13
    5 S(J)=0
  10 CONTINUE
  20 CONTINUE
  RETURN
END
```

```
ABSNT 1
ABSNT 2
ABSNT 3
ABSNT 4
ABSNT 5
ABSNT 6
ABSNT 7
ABSNT 8
ABSNT 9
ABSNT 10
ABSNT 11
ABSNT 12
ABSNT 13
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:

$$k = \frac{UBO_3 - UBO_1}{UBO_2 - 2} \quad (1)$$

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  $i = 1, 2, \dots, UBO_2$ . Then, each frequency is divided by the number of observations, n, to obtain the percent frequency:

$$P_i = \frac{100F_i}{n} \quad (2)$$

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

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

where j = selected variable

$$\text{Mean: } \bar{X} = \frac{T}{n} \quad (4)$$

Standard deviation:

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

## 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)
DIMENSION A(1),S(1),UBO(3),FREQ(1),PCT(1),STATS(5)
DIMENSION MBO(3)
DO 5 I=1,3
  MBO(I)=UBO(I)
5 CONTINUE
C CALCULATE MIN AND MAX
VMIN=1.0E38
VMAX=-1.0E38
IJ=NO(NOVAR-1)
DO 30 J=1,NO
  IF(S(J)) 10,30,10
  IF(A(IJ)-VMIN) 15,20,20
  IF(A(IJ)-VMAX) 30,30,25
  VMIN=A(IJ)
  VMAX=A(IJ)
30 CONTINUE
STATS(4)=VMIN
STATS(5)=VMAX
C DETERMINE LIMITS
IF(UBO(1)-UBO(3)) 40,35,40
35 UBO(1)=VMIN
UBO(3)=VMAX
40 INN=UBO(2)
C CLEAR OUTPUT AREAS
DO 45 I=1,INN
  FREQ(I)=0.0
  PCT(I)=0.0
45 CONTINUE
DO 50 I=1,3
  STATS(I)=0.0
50 CONTINUE
C CALCULATE INTERVAL SIZE
SINT=ABS((UBO(3)-UBO(1))/(UBO(2)-2.0))
C TEST SUBSET VECTOR
SCNT=0.0
IJ=NO(NOVAR-1)
DO 75 J=1,NO
  IF(S(J)) 55,75,55
  SCNT=SCNT+1.0
55 CONTINUE
C DEVELOP TOTAL AND FREQUENCIES
STATS(1)=STATS(1)+A(IJ)
STATS(3)=STATS(3)+A(IJ)*A(IJ)
TEMP=UBO(1)-SINT
INTX=INN-1
DO 60 I=1,INTX
  TEMP=TEMP+SINT
  IF(A(IJ)-TEMP) 70,60,60
60 CONTINUE
  IF(A(IJ)-TEMP) 75,65,65
65 FREQ(INN)=FREQ(INN)+1.0
  GO TO 75
70 FREQ(I)=FREQ(I)+1.0
75 CONTINUE
C CALCULATE RELATIVE FREQUENCIES
DO 80 I=1,INN
  PCT(I)=FREQ(I)*100.0/SCNT
80 CONTINUE
C CALCULATE MEAN AND STANDARD DEVIATION
IF(SCNT=1.0) 85,85,90
85 STATS(2)=0.0
STATS(3)=0.0
GO TO 95
90 STATS(2)=STATS(1)/SCNT
STATS(3)=SQRT(ABS((STATS(3)-STATS(1)*STATS(1)/SCNT)/(SCNT-1.0)))
95 DO 100 I=1,3
  UBO(I)=MBO(I)
100 CONTINUE
RETURN
END

```

## 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  $UBO_{1j} = UBO_{3j}$ , the subroutine finds and uses the minimum and maximum values of the  $j^{\text{th}}$  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  $i = 1, 2, \dots, UBO_{21}$

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

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:

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

Standard deviation:

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

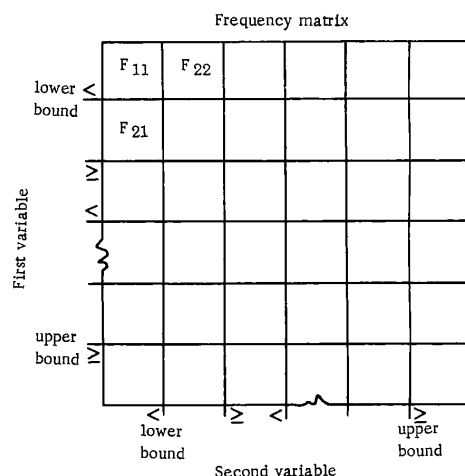


Figure 5. Frequency matrix

## 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,UPO,FREQ,PCT,STAT1,STAT2,NO,NV)
DIMENSION A(1),S(1),NOV(2),UBO(3,2),FREQ(1),PCT(1),STAT1(1),
1STAT2(2),SINT(2)
DIMENSION WBO(3,2)
DO 5 I=1,3
DO 5 J=1,2
5 WBO(I,J)=UBO(I,J)
C
DETERMINE LIMITS
DO 40 I=1,2
IF(UBO(I,1)=UBO(3,1))40,10,40
10 VMIN=1.0E38
VMAX=-1.0E38
IJ=NO*(NOV(I)-1)
DO 35 J=1,NO
IF(S(I)) 15,35,15
15 IF(A(IJ)=VMIN) 20,25,25
20 VMIN=A(IJ)
25 IF(A(IJ)=VMAX) 35,35,30
30 VMAX=A(IJ)
35 CONTINUE
UBO(I,1)=VMIN
UBO(3,1)=VMAX
40 CONTINUE
C
CALCULATE INTERVAL SIZE
45 DO 50 I=1,2
50 SINT(I)=ABS(UBO(3,I)-UBO(1,I))/(UBO(2,I)-2,0))
C
CLEAR OUTPUT AREAS
INT1=UBO(2,1)
INT2=UBO(2,2)
INTT=INT1*INT2
DO 55 I=1,INTT
FREQ(I)=0.0
55 PCT(I)=0.0
INTY=3*INT1
DO 60 I=1,INTY
60 STAT1(I)=0.0
INTZ=3*INT2
DO 65 I=1,INTZ
65 STAT2(I)=0.0
C
TEST SUBSET VECTOR
SCNT=0.0
INTY=INT1-1
INTX=INT2-1
IJ=NO*(NOV(1)-1)
IJK=NO*(NOV(2)-1)
DO 75 J=1,NO
IJ=IJ+1
IJK=IJK+1
IF(S(I)) 70,95,70
70 SCNT=SCNT+1.0
C
CALCULATE FREQUENCIES
TEMP1=UBO(1,1)-SINT(1)
DO 75 IY=1,INTY
TEMP1=TEMP1+SINT(1)
IF(A(IJ)-TEMP1) 80,75,75
75 CONTINUE
IY=INT1
80 IYV=3*(IY-1)+1
STAT1(IYV)=STAT1(IYV)+A(IJ)
IYV=IYV+1
STAT1(IYV)=STAT1(IYV)+1.0
IYV=IYV+1
STAT1(IYV)=STAT1(IYV)+A(IJ)*A(IJ)
TEMP2=UBO(1,2)-SINT(2)
DO 85 IY=1,INTY
TEMP2=TEMP2+SINT(2)
IF(A(IJK)-TEMP2) 90,85,85
85 CONTINUE
IX=INT2
90 IJF=INT1*(IX-1)+1
FREQ(IJF)=FREQ(IJF)+1.0
IF(S(I)) 95,105,95
STAT2(IX)=STAT2(IX)+A(IJK)
IX=IX+1
STAT2(IX)=STAT2(IX)+1.0
IX=IX+1
STAT2(IX)=STAT2(IX)+A(IJK)*A(IJK)
95 CONTINUE
C
CALCULATE PERCENT FREQUENCIES
DO 100 I=1,INTT
100 PCT(I)=FREQ(I)*100.0/SCNT
C
CALCULATE TOTALS, MEANS, STANDARD DEVIATIONS
IXY=-1
DO 120 I=1,INT1
IXY=IXY+3
ISD=IXY+1
TEMP1=STAT1(IXY)
SUM=STAT1(IXY+1)
IF(TEMP1-1.0) 120,105,110
105 STAT1(ISD)=0.0
GO TO 115
110 STAT1(ISD)=SORT(ABS((STAT1(ISD)-SUM*SUM/TEMP1)/(TEMP1-1.0)))
115 STAT1(IXY)=SUM/TEMP1
120 CONTINUE
IXX=-1
DO 140 I=1,INT2
IXX=IXX+3
ISD=IXX+1
TEMP2=STAT2(IXX)
SUM=STAT2(IXX+1)
IF(TEMP2-1.0) 140,125,130
125 STAT2(ISD)=0.0
GO TO 135
130 STAT2(ISD)=SORT(ABS((STAT2(ISD)-SUM*SUM/TEMP2)/(TEMP2-1.0)))
135 STAT2(IXX)=SUM/TEMP2
140 CONTINUE
DO 150 I=1,3
DO 150 J=1,2
150 UPO(I,J)=WBO(I,J)
RETURN
END

```

## 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,NV,N)
  DIMENSION A(1),D(1),S(1)
  L=0
  LL=0
  DO 20 J=1,NV
    DO 15 I=1,NO
      IF(S(I)) 15, 15, 10
    10 LL=LL+1
    D(LL)=A(I)
  15 CONTINUE
  20 CONTINUE
  C      COUNT NON-ZERO CODES IN VECTOR S
  N=0
  DO 40 I=1,NO
    IF(S(I)) 40, 30, 25
  25 N=N+1
  30 CONTINUE
  RETURN
  END
```

```
SUBMX 1
SUBMX 2
SUBMX 3
SUBMX 4
SUBMX 5
SUBMX 6
SUBMX 7
SUBMX 8
SUBMX 9
SUBMX 10
SUBMX 11
SUBMX 12
SUBMX 13
SUBMX 14
SUBMX 15
SUBMX 16
SUBMX 17
SUBMX 18
SUBMX 19
SUBMX 20
```

## Statistics - Elementary

### MOMEN

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

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

where  $UBO_1$  = given lower bound

$UBO_2$  = given class interval

$UBO_3$  = given upper bound

and the total frequency as follows:

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

where  $F_i$  = frequency count in i-th interval.

Then, the following are computed:

First Moment (Mean):

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

j-th Moment (Variance):

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

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)
  DIMENSION F(1),UBO(3),ANS(4)
  DO 100 I=1,4
    ANS(I)=0.0
  100 CONTINUE
  C   CALCULATE THE NUMBER OF CLASS INTERVALS
  N=(UBO(3)-UBO(1))/UBO(2)+0.5
  C   CALCULATE TOTAL FREQUENCY
  T=0.0
  DO 110 I=1,N
    T=T+F(I)
  110 CONTINUE
  IF(NOP=5) 130,120,115
  115 NOP=5
  120 JUMP=1
  GO TO 150
  C   FIRST MOMENT
  150 DO 160 I=1,N
    FI=F(I)
    ANS(1)=ANS(1)+FI*(UBO(1)+(FI-0.5)*UBO(2))
    ANS(1)=ANS(1)/T
    GO TO (350,200,250,300,200), NOP
  C   SECOND MOMENT
  200 DO 210 I=1,N
    FI=F(I)
    ANS(2)=ANS(2)+FI*(UBO(1)+(FI-0.5)*UBO(2)-ANS(1))**2
    ANS(2)=ANS(2)/T
    GO TO (250,350), JUMP
  C   THIRD MOMENT
  250 DO 260 I=1,N
    FI=F(I)
    ANS(3)=ANS(3)+FI*(UBO(1)+(FI-0.5)*UBO(2)-ANS(1))**3
    ANS(3)=ANS(3)/T
    GO TO (300,350), JUMP
  C   FOURTH MOMENT
  300 DO 310 I=1,N
    FI=F(I)
    ANS(4)=ANS(4)+FI*(UBO(1)+(FI-0.5)*UBO(2)-ANS(1))**4
    ANS(4)=ANS(4)/T
  310 CONTINUE
  RETURN
  END
```

```
MOMEN 1
MOMEN 2
MOMEN 3
MOMEN 4
MOMEN 5
MOMEN 6
MOMEN 7
MOMEN 8
MOMEN 9
MOMEN 10
MOMEN 11
MOMEN 12
MOMEN 13
MOMEN 14
MOMEN 15
MOMEN 16
MOMEN 17
MOMEN 18
MOMEN 19
MOMEN 20
MOMEN 21
MOMEN 22
MOMEN 23
MOMEN 24
MOMEN 25
MOMEN 26
MOMEN 27
MOMEN 28
MOMEN 29
MOMEN 30
MOMEN 31
MOMEN 32
MOMEN 33
MOMEN 34
MOMEN 35
MOMEN 36
MOMEN 37
MOMEN 38
MOMEN 39
MOMEN 40
```

## TTSTT

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

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

$$\bar{A} = \frac{\sum_{i=1}^{NA} A_i}{NA}; \quad \bar{B} = \frac{\sum_{i=1}^{NB} B_i}{NB} \quad (1)$$

and the corresponding sample variances by:

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

$\mu$  and  $\sigma^2$  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  $\mu_B$  and set  $NA = 1$  ( $A$  is stored in location  $A$ ).

The subroutine computes:

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

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

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

The subroutine computes:

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

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

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

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

The subroutine computes:

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

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

(degrees of freedom)

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

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

The subroutine computes:

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

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

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

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

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)
  DIMENSION A(1),B(1)
  C      INITIALIZATION
  NDF=0
  ANS=0.0
  C      CALCULATE THE MEAN OF A
  AMEAN=0.0
  DO 110 I=1,NA
    AMEAN=AMEAN+A(I)
  FNA=NA
  AMEAN=AMEAN/FNA
  C      CALCULATE THE MEAN OF B
  BMEAN=0.0
  DO 120 I=1,NB
    BMEAN=BMEAN+B(I)
  FNB=NB
  BMEAN=BMEAN/FNB
  IF (NOP=4) 122, 140, 200
  122 IF (NOP=1) 200, 130, 125
  C      CALCULATE THE VARIANCE OF A
  125 S2=0.0
  DO 130 I=1,NA
    S2=S2+(A(I)-AMEAN)**2
  130 S2=S2/(FNA-1.0)
  C      CALCULATE THE VARIANCE OF B
  135 S2=0.0
  DO 140 I=1,NB
    S2=S2+(B(I)-BMEAN)**2
  140 S2=S2/(FNB-1.0)
  C      OPTION 1
  150 ANS=(BMEAN-AMEAN)/SQRT(S2)*SQRT(FNB)
  NDF=NB-1
  GO TO 200
  C      OPTION 2
  160 NDF=NA+NB-2
  FNDP=V3F
  S=SQRT((FNA-1.0)*S2*(FNB-1.0)*S2)/FNDP
  ANS=(FNA*AMEAN-AMFAN/S)*F1.0/SQRT(1.0/FNA+1.0/FNB)
  GO TO 200
  C      OPTION 3
  170 ANS=(BMEAN-AMEAN)/SQRT(S2*(FNA+S2/FNB))
  A1=(V2/FNA+S2/FNB)**2
  A2=(S2/FNA)**2/(FNA+1.0)+S2/FNB**2/(FNB+1.0)
  NDF=A1/A2-2.0+0.5
  GO TO 200
  C      OPTION 4
  180 S2=0.0
  DO 190 I=1,NB
    S2=S2+(B(I)-A(I))**2
  190 S2=S2/(FNB-1.0)
  ANS=(S2/S2)*SQRT(FNB)
  NDF=FNB-1
  200 41 TIME
  END

```

## Statistics - Correlation

### 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  $i = 1, 2, \dots, n$  implies observations and  $j = 1, 2, \dots, m$  implies variables.

The following equations are used to calculate these statistics:

Sums of cross-products of deviations:

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

where  $j = 1, 2, \dots, m$ ;  $k = 1, 2, \dots, m$

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

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

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

where  $j = 1, 2, \dots, m$

Correlation coefficients:

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

where  $j = 1, 2, \dots, m$ ;  $k = 1, 2, \dots, m$

Standard deviations:

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

where  $j = 1, 2, \dots, m$

### Subroutine CORRE

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)
DIMENSION X(1),XBAR(1),STD(1),RX(1),R(1),B(1),D(1),T(1)
C INITIALIZATION
DO 100 J=1,M
  B(J)=0.0
  T(J)=0.0
  K=(M+M+1)/2
DO 102 I=1,K
  R(I)=0.0
  FN=M
  L=0
  IF(I0) 105, 127, 105
C DATA ARE ALREADY IN CORE
DO 105 J=1,M
  DO 107 I=1,M
    L=L+1
    T(J)=T(J)+X(I)
    XBAR(J)=T(J)/FN
    DO 115 I=1,M
      JK=0
      L=L+N
      DO 110 J=1,M
        D(J)=X(I)-T(J)
        B(J)=B(J)+D(J)
        DO 115 K=1,J
          JK=JK+1
          R(JK)=R(JK)+D(J)*D(K)
          GO TO 205
C READ OBSERVATIONS AND CALCULATE TEMPORARY
C MEANS FROM THESE DATA IN T(J)
DO 127 IF(N=M) 130, 130, 135
DO 130 KK=N
GO TO 137
DO 135 K=M
DO 137 DO 140 I=1,KK
  CALL DATA (M,D)
  DO 140 J=1,M
    T(J)=T(J)+D(J)
    L=L+1
DO 140 RX(L)=D(J)
  FEK=K
DO 150 J=1,M
  XBAR(J)=T(J)/FEK
C CALCULATE SUMS OF CROSS-PRODUCTS OF DEVIATIONS
C FROM TEMPORARY MEANS FOR M OBSERVATIONS
DO 150 I=1,KK
  JK=0
DO 170 J=1,M
  L=L+1
DO 170 D(J)=RX(L)-T(J)
DO 180 J=1,M
  B(J)=B(J)+D(J)
DO 180 K=1,J
  JK=JK+1
DO 180 R(JK)=R(JK)+D(J)*D(K)
IF(N=K) 205, 205, 185
C READ THE REST OF OBSERVATIONS ONE AT A TIME, SUM
C THE OBSERVATION, AND CALCULATE SUMS OF CROSS-
C PRODUCTS OF DEVIATIONS FROM TEMPORARY MEANS
DO 185 KK=N+KK
DO 200 I=1,KK
  JK=0
  CALL DATA (M,D)
  DO 190 J=1,M
    XBAR(J)=XBAR(J)+D(J)
    T(J)=T(J)+D(J)
    B(J)=B(J)+D(J)
    DO 200 J=1,M
      DO 200 K=1,J
        JK=JK+1
        R(JK)=R(JK)+D(J)*D(K)
C CALCULATE MEANS
DO 205 JK=0
DO 210 J=1,M
  XBAR(J)=XBAR(J)/FN
C ADJUST SUMS OF CROSS-PRODUCTS OF DEVIATIONS
C FROM TEMPORARY MEANS
DO 210 K=1,J
  JK=JK+1
DO 210 R(JK)=R(JK)+D(J)*D(K)/FN
C CALCULATE CORRELATION COEFFICIENTS
DO 220 J=1,M
  JK=JK+J
  STD(J)=SQRT(ABS(R(JK)))
DO 230 J=1,M
  DO 230 K=J+M
  L=L+(K-J)/2
  RX(L)=R(JK)
  L=L+(K-1)+J

```

```

RX(L)=R(JK)
IF(STD(J)*STD(K)) 225, 222, 225
222 R(JK)=0.0
GO TO 230
225 R(JK)=R(JK)/(STD(J)*STD(K))
C CONTINUE
C CALCULATE STANDARD DEVIATIONS
FN=SQRT(FN-1.0)
DO 240 J=1,M
240 STD(J)=STD(J)/FN
C COPY THE DIAGONAL OF THE MATRIX OF SUMS OF CROSS-PRODUCTS OF
C DEVIATIONS FROM MEANS.
L=M
DO 250 I=1,M
  L=L+M+1
250 B(I)=RX(L)
RETURN
END
CORRE 97
CORREMO1
CORREMO2
CORREMO3
CORREMO4
CORREMO5
CORREMO6
CORREMO7
CORREMO8
CORREMO9
CORREMO10
CORREMO11
CORREMO12
CORREMO13
CORREMO14
CORREMO15
CORREMO16
CORREMO17
CORREMO18
CORREMO19
CORREMO20
CORREMO21
CORREMO22
CORREMO23
CORREMO24
CORREMO25
CORREMO26
CORREMO27
CORREMO28
CORREMO29
CORREMO30
CORREMO31
CORREMO32
CORREMO33
CORREMO34
CORREMO35
CORREMO36
CORREMO37
CORREMO38
CORREMO39
CORREMO40
CORREMO41
CORREMO42
CORREMO43
CORREMO44
CORREMO45
CORREMO46
CORREMO47
CORREMO48
CORREMO49
CORREMO50
CORREMO51
CORREMO52
CORREMO53
CORREMO54
CORREMO55
CORREMO56
CORREMO57
CORREMO58
CORREMO59
CORREMO60
CORREMO61
CORREMO62
CORREMO63
CORREMO64
CORREMO65
CORREMO66
CORREMO67
CORREMO68
CORREMO69
CORREMO70
CORREMO71
CORREMO72
CORREMO73
CORREMO74
CORREMO75
CORREMO76
CORREMO77
CORREMO78
CORREMO79
CORREMO80
CORREMO81
CORREMO82
CORREMO83
CORREMO84
CORREMO85
CORREMO86
CORREMO87
CORREMO88
CORREMO89
CORREMO90
CORREMO91
CORREMO92
CORREMO93
CORREMO94
CORREMO95
CORREMO96

```

#### 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)
DIMENSION R(11),ISAVE(11),RX(11),RY(11)
C
C   COPY INTERCORRELATIONS OF INDEPENDENT VARIABLES
C   WITH DEPENDENT VARIABLE
      MM=0
      DO 130 J=1,K
        L2=ISAVE(J)
        IF(NDEP=L2) 122, 123, 123
        L1=NDEP+L2-1
        GO TO 125
      122 L=MM+L2-L1
      123 L=L2+(NDEP-NDEP-VDEP)/2
      125 RY(L)=R(L)
C
C   COPY A SUBSET MATRIX OF INTERCORRELATIONS AMONG
C   INDEPENDENT VARIABLES
      DO 130 I=1,K
        L1=ISAVE(I)
        IF(L1=L2) 127, 128, 128
        L=L1+L2-L1
        GO TO 129
      127 L=L2+(L1-L1)/2
      128 L=L2+(L1-L1)/2
      129 MM=MM+1
      130 RX(MM)=R(L)
C
C   PLACE THE SUBSCRIPT NUMBER OF THE DEPENDENT
C   VARIABLE IN ISAVE(K+1)
      ISAVE(K+1)=NDEP
      RETURN
      END
```

## 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  $i^{\text{th}}$  independent variable with dependent variable

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

$i, j = 1, 2, \dots, k$  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  $j^{\text{th}}$  independent variable

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

$s_y$  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  $j^{\text{th}}$  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:

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

where  $n$  = number of observations

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

Standard deviations of regression coefficients:

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

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

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

Computed  $t$ :

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

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

## 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)
DIMENSION XBAR(1),STD(1),D(1),RX(1),RY(1),ISAVE(1),B(1),SB(1),
1 T(1),ANS(10)
MULTR 1
1 N=K+1
MULTR 2
      BETA WEIGHTS
MULTR 3
      DO 100 J=1,K
MULTR 4
      B(J)=0.0
MULTR 5
      DO 110 J=1,K
MULTR 6
      L1=K*(J-1)
MULTR 7
      DO 110 I=1,K
MULTR 8
      L=L1+I
MULTR 9
      L1=L+I
MULTR 10
      B(J)=B(J)+RY(I)*RX(L)
MULTR 11
      RM=0.0
MULTR 12
      DO 120 I=1,K
MULTR 13
      B(J)=B(J)+STD(I)*STD(L)
MULTR 14
      RM=RM+D(I)
MULTR 15
      L1=ISAVE(I)
MULTR 16
      COEFFICIENT OF DETERMINATION
MULTR 17
      DO 120 I=1,K
MULTR 18
      RM=RM+D(I)*RY(I)
MULTR 19
      REGRESSION COEFFICIENTS
MULTR 20
      L=ISAVE(1)
MULTR 21
      B(1)=B(1)+STD(L)/STD(L1)
MULTR 22
      INTERCEPT
MULTR 23
      DO 120 I=1,K
MULTR 24
      B(I)=B(I)+XBAR(L1)-B(1)*XBAR(L)
MULTR 25
      SUM OF SQUARES ATTRIBUTABLE TO REGRESSION
MULTR 26
      SSAR=RM*D(L1)
MULTR 27
      MULTIPLE CORRELATION COEFFICIENT
MULTR 28
      C 122 RM= SQRT(ABS(RM))
MULTR 29
      SUM OF SQUARES OF DEVIATIONS FROM REGRESSION
MULTR 30
      SSDR=D(L1)-SSAR
MULTR 31
      VARIANCE OF ESTIMATE
MULTR 32
      FN=N-K-1
MULTR 33
      SY=SSDR/FN
MULTR 34
      STANDARD DEVIATIONS OF REGRESSION COEFFICIENTS
MULTR 35
      DO 130 J=1,K
MULTR 36
      L1=K*(J-1)+J
MULTR 37
      L=ISAVE(L1)
MULTR 38
      SRTJ1= SQRT(ABS(RX(L1)/D(L1)*SY))
MULTR 39
      COMPUTED T-VALUES
MULTR 40
      C 130 T(J)=B(J)/SRTJ1
MULTR 41
      STANDARD ERROR OF ESTIMATE
MULTR 42
      C 135 SY= SQRT(ABS(SY))
MULTR 43
      F VALUE
MULTR 44
      FK=K
MULTR 45
      SSAR=SSAR/FK
MULTR 46
      SSOR=SSDR/FN
MULTR 47
      F=SSAR/SSOR
MULTR 48
      ANS(1)=B(1)
MULTR 49
      ANS(2)=RM
MULTR 50
      ANS(3)=SY
MULTR 51
      ANS(4)=SSAR
MULTR 52
      ANS(5)=FK
MULTR 53
      ANS(6)=SSOR
MULTR 54
      ANS(7)=F
MULTR 55
      ANS(8)=FN
MULTR 56
      ANS(9)=F
MULTR 57
      RETURN
MULTR 58
      END
MULTR 59
```

## Statistics - Polynomial Regression

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

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

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,  $b_0, b_1, b_2, \dots, b_m$ , 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  $m^{\text{th}}$  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  $i^{\text{th}}$  case of the independent variable;

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

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

$n$  - number of cases (observations)

$p = m + 1$

$m$  = highest degree polynomial specified

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

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

where  $i$  and  $m$  are as defined as above.

Then, the following are calculated:

Means:

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

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

Sums of cross-products of deviations from means:

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

where  $j = 1, 2, \dots, p$ ;  $k = 1, 2, \dots, p$ .

Correlation coefficients:

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

where  $i = 1, 2, \dots, p$ ;  $j = 1, 2, \dots, p$ .

Standard deviations:

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

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

#### Subroutine GDATA

Purpose:

Generate independent variables up to the  $M^{\text{th}}$  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 inde-

pendent 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)
  DIMENSION K(1),XBAR(1),STD(1),D(1),SUMSQ(1)
  GENERATE INDEPENDENT VARIABLES
  IF (M-1) 105,105,90
90  L1=0
  DO 170 J=2,M
    L1=L1+M
    DO 100 J=1,N
      L=L1+J
      K(L)=X(K1)*X(J)
100  K(L)=X(K1)*X(J)
  C  CALCULATE MEANS
  DO 105 M=1
    M=M+1
    L=M
    DO 115 J=1,M
      XBAR(L)=XBAR(L)+X(J)
      DO 110 J=1,N
        L=L+1
110  XBAR(L)=XBAR(L)+X(J)
115  XBAR(L)=XBAR(L)/N
    DO 130 J=1,M
      STD(J)=0.0
130  STD(J)=0.0
  C  CALCULATE SUMS OF CROSS-PRODUCTS OF DEVIATIONS
  L=((M+1)*M+1)/2
  DO 150 I=1,L
150  D(I)=0.0
    DO 170 K=1,N
      L=0
      DO 170 J=1,M
        L2=M*(J-1)+K
        T2=X(L2)-XBAR(J)
        STD(J)=STD(J)+T2
        DO 170 I=1,J
          L1=M*(I-1)+K
          T1=X(L1)-XBAR(I)
          L=L+1
170  D(I)=D(I)+T1*T2
      L=L+1
      DO 175 J=1,M
        DO 175 I=1,J
          L=L+1
175  D(I)=D(I)-STD(I)*STD(J)/N
      L=L+1
      DO 180 I=1,M
        L=L+I
        SUMSQ(L)=0.0
180  STD(L)=STD(L)+X(L)
  C  CALCULATE CORRELATION COEFFICIENTS
  L=0
  DO 190 J=1,M
    DO 190 I=1,J
      L=L+1
190  D(L)=D(L)/(STD(I)*STD(J))
  C  CALCULATE STANDARD DEVIATIONS
  DO 200 J=1,M
    STD(J)=STD(J)/N
200  STD(J)=STD(J)/N
  END

```

## 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:

$$R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \quad (1)$$

$R_{11}$  = 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:

$$\begin{vmatrix} R_{22}^{-1} R_{21} R_{11}^{-1} R_{12} - \lambda I \end{vmatrix} = 0 \quad (2)$$

is then solved for all values of  $\lambda$ , 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 ( $\lambda_i$ ) with associated eigenvectors of  $R_{22}^{-1}A$ , where  $i = 1, 2, \dots, q$ .

For each subscript  $i = 1, 2, \dots, q$ , the following statistics are calculated:

Canonical correlation:

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

where  $\lambda_i$  =  $i^{\text{th}}$  eigenvalue

Chi-square:

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

where  $n$  = number of observations

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

Degrees of freedom for  $\chi^2$ :

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

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

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

where  $v_{ki}$  = eigenvector associated with  $\lambda_i$

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

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

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

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

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

#### 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,COEFL,R)
  DIMENSION RRE(1),ROOTS(1),WLAM(1),CANR(1),CHISQ(1),NDF(1),COEFR(1),COEFL(1)
  C 1 COEFL(1)=R(1)
  C 2 PARTITION INTERCORRELATIONS AMONG LEFT HAND VARIABLES, BETWEEN
  C 3 LEFT AND RIGHT HAND VARIABLES, AND AMONG RIGHT HAND VARIABLES.
  C 4
  N1=1
  DO 105 J=1,M
    DO 105 J=1,M
      IF (J-J) 102, 103, 103
    102 L=1+(J+J-1)/2
    GO TO 104
    103 L=J*(J+1)/2
    104 N1=N1+1
    105 K(1)=R(1)
    L=MP
    DO 108 J=2,M
      N1=N1+J-1
      DO 108 I=1,M
        L=L+1
        N1=N1+1
    108 R(1)=R(N1)
    N2=MP+1
    L=0
    DO 110 J=N2,M
      N1=N1+J-1
      DO 110 I=1,M
        L=L+1
        N1=N1+1
    110 COEFL(1)=R(N1)
    L=0
    DO 120 J=N2,M
      N1=N1+J-1+MP
      DO 120 I=N2,M
        L=L+1
        N1=N1+1
    120 COEFR(1)=R(N1)
  C 121 SOLVE THE CANONICAL EQUATION
  C 122 L=MP+MP+1
  C 123 K=L+MP
  C 124 CALL MINV (R,MP,DET,R(1),R(1))
  C 125 CALCULATE T = INVERSE OF R11 * R12
  DO 140 J=1,MP
    N2=0
    DO 130 J=1,MQ
      N1=1+MP
      R(1)=R(N1)
      DO 130 K=1,M
        N1=N1+MP
        N2=N2+1
    130 ROOTS(J)=R(1)+R(N1)*COEFL(N2)
    L=L+MP
    DO 140 J=1,MQ

```

```

      L=L*MP
140 R(L)=ROOTS(J)
      C      CALCULATE A = R21 * T
      L=MP*MQ
      N3=L+1
      DO 160 J=1,MQ
      N1=0
      DO 160 I=1,MQ
      N2=MP*(J-I)
      SUM=0.0
      DO 150 K=1,MP
      N1=N1+1
      N2=N2+1
150 SUM=SUM+COEFL(N1)*R(N2)
      L=L+1
160 R(L)=SUM
      C      CALCULATE EIGENVALUES WITH ASSOCIATED EIGENVECTORS OF THE
      C      INVERSE OF R22 * A
      L=L+1
      CALL NROOT (MQ,R(N3),COEFL,ROOTS,R(L))
      C      FOR EACH VALUE OF I = 1, 2, ..., MQ, CALCULATE THE FOLLOWING
      C      STATISTICS
      DO 210 I=1,MQ
      C      TEST WHETHER EIGENVALUE IS GREATER THAN ZERO
      IF(ROOTS(I)) 220, 220, 165
      C      CANONICAL CORRELATION
165 CANR(I)= SORT(ROOTS(I))
      C      CHI-SQUARE
      WLAH(I)=1.0
      DO 170 J=1,MQ
170 WLAH(I)=WLAH(I)*(1.0-ROOTS(J))
      FNP=N
      FNP=MP
      FNP=MQ
      BAT = WLAH(I)
175 CHISQ(I) = -(FNP-3.5*(FNP+FNP+1.0))*ALOG(BAT)
      C      DEGREES OF FREEDOM FOR CHI-SQUARE
      N1=I-1
      NDF(I)=(MP-N1)*(MQ-N1)
      C      I-TH SET OF RIGHT HAND COEFFICIENTS
      N1=MQ*(I-1)
      N2=MQ*(I-1)+1
      DO 180 J=1,MQ
      N1=N1+1
      N2=N2+1
180 COEFL(N1)=R(N2)
      C      I-TH SET OF LEFT HAND COEFFICIENTS
      DO 200 J=1,MP
      N1=J-MP
      N2=MQ*(I-1)
      K=MP*(I-1)+J
      COEFL(K)=0.0
      DO 190 JJ=1,MQ
      N1=N1+MP
      N2=N2+1
190 COEFL(K)=COEFL(K)+R(N1)*COEFL(N2)
200 COEFL(K)=COEFL(K)/CANR(I)
210 CONTINUE
220 RETURN
      END

```

```

CANOR 55
CANOR 56
CANOR 57
CANOR 58
CANOR 59
CANOR 60
CANOR 61
CANOR 62
CANOR 63
CANOR 64
CANOR 65
CANOR 66
CANOR 67
CANOR 68
CANOR 69
CANOR 70
CANOR 71
CANOR 72
CANOR 73
CANOR 74
CANOR 75
CANOR 76
CANOR 77
CANOR 78
CANOR 79
CANOR 80
CANOR 81
CANOR 82
CANOR 83
CANOR 84
CANOR 85
CANOR 86
CANOR 87
CANOR 88
CANOR 89
CANOR 90
CANOR 91
CANOR 92
CANOR 93
CANOR 94
CANOR 95
CANOR 96
CANOR 97
CANOR 98
CANOR 99
CANOR100
CANOR101
CANOR102
CANOR103
CANOR104
CANOR105
CANOR106
CANOR107
CANOR108
CANOR109
CANOR110
CANOR111
CANOR112
CANOR113
CANOR114

```

## NROOT

This subroutine calculates the eigenvalues,  $\lambda_i$ , 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:

$$\mu_i = \frac{1}{\sqrt{h_i}} \quad (1)$$

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

$m$  = order of matrix  $B$

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

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

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

$$S = QB^{-1/2} \quad (3)$$

and eigenvalues,  $\lambda_i$ , 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:

$$V_{ij} = \frac{W_{ij}}{\sqrt{\text{SUM}V_j}} \quad (4)$$

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

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

$$\text{SUM}V_j = \sum_{i=1}^m W_{ij}^2 \quad (5)$$

## 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.

```
110 XL(J)=1.0/ SQRT( ABS(XL(J)) )
      K=0
      DO 115 J=1,M
      DO 115 I=1,M
      K=K+1
115 B(K)=X(K)*XL(J)
C      FORM (B**(-1/2))PRIME * A * (B**(-1/2))
      DO 120 I=1,M
      N2=0
      DO 120 J=1,M
      N1=N1+(-1)
      L=N1*(J-1)+1
      XL(I)=0.0
      DO 120 K=1,M
      A1=N1+1
      N2=N2+1
120 XL(I)=XL(I)+R(N1)*A(N2)
      L=0
      DO 130 J=1,M
      DO 130 I=1,J
      N1=I-M
      N2=N1*(J-1)+1
      L=L+1
      A(I)=0.0
      DO 130 K=1,M
      N1=N1+M
      N2=N2+1
130 A(I)=A(I)+X(N1)*B(N2)
C      COMPUTE EIGENVALUES AND EIGENVECTORS OF A
      CALL EIGEN (A,X,M,MV)
      L=0
      DO 140 I=1,M
      L=L+1
140 XL(I)=A(I)
C      COMPUTE THE NORMALIZED EIGENVECTORS
      DO 150 I=1,M
      N2=0
      DO 150 J=1,M
      N1=I-M
      L=N1*(J-1)+1
      A(I)=0.0
      DO 150 K=1,M
      N1=N1+M
      N2=N2+1
150 A(I)=A(I)+R(N1)*X(N2)
      L=0
      K=0
      DO 160 J=1,M
      SUMV=0.0
      DO 170 I=1,M
      L=L+1
170 SUMV=SUMV+A(I)*A(I)
175 SUMV= SQRT(SUMV)
      DO 180 I=1,M
      K=K+1
180 X(K)=A(I)/SUMV
      RETURN
      END
```

NROOT 19  
NROOT 20  
NROOT 21  
NROOT 22  
NROOT 23  
NROOT 24  
NROOT 25  
NROOT 26  
NROOT 27  
NROOT 28  
NROOT 29  
NROOT 30  
NROOT 31  
NROOT 32  
NROOT 33  
NROOT 34  
NROOT 35  
NROOT 36  
NROOT 37  
NROOT 38  
NROOT 39  
NROOT 40  
NROOT 41  
NROOT 42  
NROOT 43  
NROOT 44  
NROOT 45  
NROOT 46  
NROOT 47  
NROOT 48  
NROOT 49  
NROOT 50  
NROOT 51  
NROOT 52  
NROOT 53  
NROOT 54  
NROOT 55  
NROOT 56  
NROOT 57  
NROOT 58  
NROOT 59  
NROOT 60  
NROOT 61  
NROOT 62  
NROOT 63  
NROOT 64  
NROOT 65  
NROOT 66  
NROOT 67  
NROOT 68  
NROOT 69  
NROOT 70  
NROOT 71  
NROOT 72  
NROOT 73  
NROOT 74  
NROOT 75  
NROOT 76

```
SUBROUTINE NROOT (M,A,B,XL,X)
      DIMENSION A(1),B(1),XL(1),X(1)
C      COMPUTE EIGENVALUES AND EIGENVECTORS OF B
      K=1
      DO 100 J=2,M
      L=N1*(J-1)+1
      DO 100 I=1,J
      L=L+1
      K=K+1
100 B(K)=B(I)
C      THE MATRIX B IS A REAL SYMMETRIC MATRIX.
      MV=0
      CALL EIGEN (B,X,M,MV)
C      FORM RECIPROCAL OF SQUARE ROOT OF EIGENVALUES. THE RESULTS
C      ARE PMULTIPLIED BY THE ASSOCIATED EIGENVECTORS.
      L=0
      DO 110 J=1,M
      L=L+J
      NROOT 1
      NROOT 2
      NROOT 3
      NROOT 4
      NROOT 5
      NROOT 6
      NROOT 7
      NROOT 8
      NROOT 9
      NROOT 10
      NROOT 11
      NROOT 12
      NROOT 13
      NROOT 14
      NROOT 15
      NROOT 16
      NROOT 17
      NROOT 18
```

## 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:

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

where  $L_i$  = number of levels of  $i^{\text{th}}$  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:

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

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

where  $J = 2, 3, \dots, k$

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

$$S = \text{KOUNT}_1 + \sum_{j=2}^k s_j \cdot (\text{KOUNT}_j - 1) \quad (4)$$

where  $\text{KOUNT}_j$  = value of  $j^{\text{th}}$  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  $X = (\text{LEVEL}(1) + 1) * (\text{LEVEL}(2) + 1) * \dots * (\text{LEVEL}(K) + 1)$ .)
- 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. Wulf, John Wiley and Sons, 1962, Chapter 20.

```

SUBROUTINE AVDAT (K,LEVEL,N,K,L,ISTEP,KOUNT)
  DIMENSION LEVEL(1),X(1),ISTEP(1),KOUNT(1)
  C      CALCULATE TOTAL DATA AREA REQUIRED
  M=LEVEL(1)+1
  DO 105 I=2,K
    105 N=N*(LEVEL(I)+1)
  C      MOVE DATA TO THE UPPER PART OF THE ARRAY X
  C      FOR THE PURPOSE OF REARRANGEMENT
  N1=N+1
  N2=N+1
  DO 107 I=1,N
    N1=N1-1
    N2=N2-1
  107 X(N1)=X(N2)
  C      CALCULATE MULTIPLIERS TO BE USED IN FINDING STORAGE LOCATIONS
  C      FOR INPUT DATA
  ISTEP(1)=1
  DO 110 I=2,K
    110 ISTEP(I)=ISTEP(I-1)*(LEVEL(I-1)+1)
  DO 115 I=1,K
    115 KOUNT(I)=1
  C      PLACE DATA IN PROPER LOCATIONS
  N1=N1-1
  DO 135 I=1,N
    L=KOUNT(1)
    DO 120 J=2,K
      120 L=L+ISTEP(J)*(KOUNT(J)-1)
      N1=N1+1
      X(L)=X(N1)
      DO 130 J=1,K
        IF (KOUNT(J)-LEVEL(J)) 124, 125, 124
      124 KOUNT(J)=KOUNT(J)+1
      GO TO 135
    125 KOUNT(J)=1
  135 CONTINUE
  135 CONTINUE
  RETURN
  END

```

AVCAL

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

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

With regard to the first factor A,

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

operator  $\Delta_a =$  multiply all items by A and subtract the result of  $\sum_a$  from all items

In mathematical notations, these operators are defined as follows:

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

$$\Delta_a x_{abc} \equiv A x_{abc} - X_{.bc} \quad (2)$$

The operators  $\Sigma$  and  $\Delta$  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)
  DIMENSION LEVEL(1),X(1),ISTEP(1),LASTS(1)
  C CALCULATE THE LAST DATA POSITION OF EACH FACTOR
  LASTS(1)=L+1
  DO 145 I=2,K
    LASTS(I)=LASTS(I-1)+ISTEP(I)
  C PERFORM CALCULUS OF OPERATION
  150 DO 175 I=1,K
    L=1
    LL=1
    SUM=0.0
    NN=LEVEL(I)
    FNO=NN
    INCR=ISTEP(I)
    LAST=LASTS(I)
  C SIGMA OPERATION
  155 DO 160 J=1,NN
    SUM=SUM+X(L)
    160 L=L+INCR
    X(L)=SUM
  C DELTA OPERATION
  165 J=1,NN
    KELL=K*FNO*LL-1-SJM
    165 LL=LL+INCR
    SUM=0.0
    IF(L-LAST) 167, 175, 175
    167 IF(L-LAST+INCR) 169, 169, 175
    169 L=L+INCR
    LL=LL+INCR
    GU=1
    170 L=L+INCR+1-LAST
    LL=LL+INCR+1-LAST
    GU=1
    175 CONTINUE
  RETURN
  ENDS
```

#### 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  $\Sigma$  and  $\Delta$  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) <sup>2</sup>                                     | ABC·A                                                           | (A-1)                                            |
| (B) <sup>2</sup>                                     | ABC·B                                                           | (B-1)                                            |
| (AB) <sup>2</sup>                                    | ABC·AB                                                          | (A-1)(B-1)                                       |
| (C) <sup>2</sup>                                     | ABC·C                                                           | (C-1)                                            |
| (AC) <sup>2</sup>                                    | ABC·AC                                                          | (A-1)(C-1)                                       |
| (BC) <sup>2</sup>                                    | ABC·BC                                                          | (B-1)(C-1)                                       |
| (ABC) <sup>2</sup>                                   | 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  $(LEVEL(1) + 1) * (LEVEL(2) + 1) * \dots * (LEVEL(K) + 1)$ .
- 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 MEANO (K,LEVEL,X,GMEAN,SUMSQ,NDF,SMEAN,MSTEP,KOUNT,
1 LASTS)
1 DIMENSION LEVEL(1),X(1),SUMSQ(1),NDF(1),SMEAN(1),MSTEP(1),
1 KOUNT(1),LASTS(1)
C CALCULATE TOTAL NUMBER OF DATA
N=LEVEL(1)
DO 150 I=2,K
150 N=N*LEVEL(I)
C SET UP CONTROL FOR MEAN SQUARE OPERATOR
LASTS(1)=LEVEL(1)
DO 178 I=2,K
178 LASTS(I)=LEVEL(I)+1
NN=1
C CLEAR THE AREA TO STORE SUMS OF SQUARES
LL=(2**K)-1
MSTEP(1)=1
DO 180 I=2,K
180 MSTEP(I)=MSTEP(I-1)*2
DO 185 I=1,LL
185 SUMSQ(I)=0.0
C PERFORM MEAN SQUARE OPERATOR
DO 190 I=1,K
190 KOUNT(I)=0
DO 200 I=1,K
200 L=0
DO 260 J=1,K
IF(KOUNT(I)=LASTS(I)) 210, 250, 210
210 IF(I) 220, 220, 240
220 KOUNT(I)=KOUNT(I)+1
IF(KOUNT(I)=LEVEL(I)) 230, 230, 250
230 L=L+MSTEP(I)
GO TO 260
240 IF(KOUNT(I)=LEVEL(I)) 230, 260, 230
250 KOUNT(I)=0
260 CONTINUE
IF(I) 285, 285, 270
270 SUMSQ(L)=SUMSQ(L)+X(NN)*X(NN)
NN=NN+1
GO TO 200
C CALCULATE THE GRAND MEAN
285 FN=N
GMEAN=X(NN)/FN
C CALCULATE FIRST DIVISOR REQUIRED TO FORM SUM OF SQUARES AND
C DIVISOR WHICH IS EQUAL TO DEGREES OF FREEDOM, REQUIRED TO FORM MEAN

```

```

C MEAN SQUARES
DO 310 I=2,K
310 MSTEP(I)=0
NN=0
MSTEP(1)=1
320 ND1=1
ND2=1
DO 340 I=1,K
IF(MSTEP(I)) 330, 340, 330
330 ND1=ND1*LEVEL(I)
ND2=ND2*(LEVEL(I)-1)
340 CONTINUE
FN1=ND1
FN2=ND2
NN=NN+1
SUMSQ(NN)=SUMSQ(NN)/FN1
NDF(NN)=ND2
SMEAN(NN)=SUMSQ(NN)/FN2
IF(NN=LL) 345, 370, 370
345 DO 360 I=1,K
IF(MSTEP(I)) 347, 350, 347
347 MSTEP(I)=0
GO TO 360
350 MSTEP(I)=1
GO TO 320
360 CONTINUE
370 RETURN
END

```

```

MEANO 44
MEANO 45
MEANO 46
MEANO 47
MEANO 48
MEANO 49
MEANO 50
MEANO 51
MEANO 52
MEANO 53
MEANO 54
MEANO 55
MEANO 56
MEANO 57
MEANO 58
MEANO 59
MEANO 60
MEANO 61
MEANO 62
MEANO 63
MEANO 64
MEANO 65
MEANO 66
MEANO 67
MEANO 68
MEANO 69
MEANO 70
MEANO 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  $k = 1, 2, \dots, g$ , the subroutine calculates means and sums of cross-products of deviations from means as follows:

Means:

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

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

$j = 1, 2, \dots, m$  are variables

Sum of cross-products of deviations from means:

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

where  $j = 1, 2, \dots, m$

$\ell = 1, 2, \dots, m$

The pooled dispersion matrix is calculated as follows:

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

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  $T = N(1) + N(2) + \dots + N(K)$ .
- 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)
DIMENSION N(1),X(1),XBAR(1),D(1),CMEAN(1)
MM=MM+M
DO 100 I=1,MM
100 D(I)=0.0
C      CALCULATE MEANS
  N4=0
  L=0
  LM=0
  DO 160 NG=1,K
    N1=N(NG)
    FM=N1
    DO 130 J=1,M
      LM=LM+1
      XBAR(LM)=0.0
      DO 120 I=1,N1
        L=L+1
        XBAR(LM)=XBAR(LM)+X(I)
120 XBAR(LM)=XBAR(LM)/FM
130 XBAR(LM)=XBAR(LM)/N1
C      CALCULATE SUMS OF CROSS-PRODUCTS OF DEVIATIONS
  LMEAN=LM-M
  DO 150 I=1,N1
    LL=NA+I-N1
    DO 140 J=1,M
      LL=LL+1
      NZ=LMEAN+J
      140 CMEAN(J)=X(ILL)-XBAR(NZ)
      LL=0
      DO 150 J=1,M
        DO 150 JJ=1,M
          LL=LL+1
          150 D(ILL)=D(ILL)+CMEAN(J)*CMEAN(JJ)
160 NA=NA+N1*M
C      CALCULATE THE POOLED DISPERSION MATRIX
  LL=0
  DO 170 I=1,K
    LL=LL+N(I)
    FM=LL
    DO 180 I=1,MM
      180 D(I)=D(I)/FM
  RETURN
END
```

DMATX 1  
DMATX 2  
DMATX 3  
DMATX 4  
DMATX 5  
DMATX 6  
DMATX 7  
DMATX 8  
DMATX 9  
DMATX 10  
DMATX 11  
DMATX 12  
DMATX 13  
DMATX 14  
DMATX 15  
DMATX 16  
DMATX 17  
DMATX 18  
DMATX 19  
DMATX 20  
DMATX 21  
DMATX 22  
DMATX 23  
DMATX 24  
DMATX 25  
DMATX 26  
DMATX 27  
DMATX 28  
DMATX 29  
DMATX 30  
DMATX 31  
DMATX 32  
DMATX 33  
DMATX 34  
DMATX 35  
DMATX 36  
DMATX 37  
DMATX 38  
DMATX 39  
DMATX 40  
DMATX 41  
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

$j = 1, 2, \dots, m$  are variables

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

$\bar{x}_{jk}$  = mean of  $j^{\text{th}}$  variable in  $k^{\text{th}}$  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  $k^* = 1, 2, \dots, g$ , the following statistics are calculated:

Coefficients:

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

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

$k = k^*$

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

Discriminant functions:

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

where  $k^* = 1, 2, \dots, g$

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  $T = N(1) + N(2) + \dots + N(K)$ .

- 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 ( $T = N(1) + N(2) + \dots + N(K)$ ).
- 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,K,XBAR,D,CMEAN,V,C,P,LG)
  DIMENSION N(1),X(1),XBAR(1),D(1),CMEAN(1),C(1),P(1),LG(1)
  C
  CALCULATE COMMON MEANS
  N1=N(1)
  DO 100 I=2,K
    N1=N1+N(I)
  100 N1=N1+N(I)
  PNT=N1
  DO 110 I=1,K
    P(I)=N(I)
  110 P(I)=N(I)
  DO 120 I=1,M
    CMEAN(I)=0
  120 CMEAN(I)=CMEAN(I)+P(I)*XBAR(I,N1)
  DISCR 1
  DISCR 2
  DISCR 3
  DISCR 4
  DISCR 5
  DISCR 6
  DISCR 7
  DISCR 8
  DISCR 9
  DISCR 10
  DISCR 11
  DISCR 12
  DISCR 13
  DISCR 14
  DISCR 15

```

```

130 CMEAN(I)=CMEAN(I)/PNT
  CALCULATE GENERALIZED MAHALANOBIS D SQUARE
  L=0
  DO 140 I=1,K
    DO 140 J=1,M
      L=L+1
  140 C(L)=XBAR(I)-CMEAN(I)
  V=0.0
  L=0
  DO 160 J=1,M
    DO 160 I=1,M
      N1=I-M
      N2=J-M
      SUM=0.0
      DO 150 IJ=1,M
        N1=N1+M
        N2=N2+M
        SUM=SUM+P(IJ)*C(N1)*C(N2)
  150 SUM=SUM+P(IJ)*C(N1)*C(N2)
  L=L+1
  160 V=V+D(L)*SUM
  C
  CALCULATE THE COEFFICIENTS OF DISCRIMINANT FUNCTIONS
  N2=0
  DO 190 KA=1,K
    DO 170 I=1,M
      N2=N2+1
  170 P(I)=XBAR(N2)
  180 SUM=SUM+D(N1)*P(I)*P(I)
  C(1)=--(SUM/2.0)
  DO 190 I=1,M
    N1=I-M
    IQ=IQ+1
    C(IQ)=0.0
    DO 190 J=1,M
      N1=N1+M
      190 C(IQ)=C(IQ)+D(N1)*P(IJ)
  C
  FOR EACH CASE IN EACH GROUP, CALCULATE..
  C
  DISCRIMINANT FUNCTIONS
  LBASE=0
  N1=0
  DO 270 KG=1,K
    NN=N(KG)
    DO 260 I=1,NN
      L1=NN+LBASE
      DO 200 J=1,M
        L=L+NN
        200 D(IJ)=X(L)
        N2=0
        DO 220 KA=1,K
          N2=N2+1
          SUM=C(N2)
          DO 210 J=1,M
            N2=N2+1
            SUM=SUM+C(N2)*D(IJ)
        210 SUM=SUM+C(N2)*D(IJ)
        220 XBAR(KA)=SUM
        C
        THE LARGEST DISCRIMINANT FUNCTION
        L=1
        SUM=XBAR(L)
        DO 240 J=2,K
          IF(SUM-XBAR(J)) 230, 240, 240
        230 L=J
        SUM=XBAR(J)
        240 CONTINUE
        PROBABILITY ASSOCIATED WITH THE LARGEST DISCRIMINANT FUNCTION
        PL=0.0
        DO 250 J=1,K
          250 PL=PL+EXP(XBAR(J)-SUM)
        N1=N1+1
        LG(N1)=L
        260 P(N1)=1.0/PL
        270 LBASE=LBASE+NN+M
        RETURN
      END
  DISCR 16
  DISCR 17
  DISCR 18
  DISCR 19
  DISCR 20
  DISCR 21
  DISCR 22
  DISCR 23
  DISCR 24
  DISCR 25
  DISCR 26
  DISCR 27
  DISCR 28
  DISCR 29
  DISCR 30
  DISCR 31
  DISCR 32
  DISCR 33
  DISCR 34
  DISCR 35
  DISCR 36
  DISCR 37
  DISCR 38
  DISCR 39
  DISCR 40
  DISCR 41
  DISCR 42
  DISCR 43
  DISCR 44
  DISCR 45
  DISCR 46
  DISCR 47
  DISCR 48
  DISCR 49
  DISCR 50
  DISCR 51
  DISCR 52
  DISCR 53
  DISCR 54
  DISCR 55
  DISCR 56
  DISCR 57
  DISCR 58
  DISCR 59
  DISCR 60
  DISCR 61
  DISCR 62
  DISCR 63
  DISCR 64
  DISCR 65
  DISCR 66
  DISCR 67
  DISCR 68
  DISCR 69
  DISCR 70
  DISCR 71
  DISCR 72
  DISCR 73
  DISCR 74
  DISCR 75
  DISCR 76
  DISCR 77
  DISCR 78
  DISCR 79
  DISCR 80
  DISCR 81
  DISCR 82
  DISCR 83
  DISCR 84
  DISCR 85
  DISCR 86
  DISCR 87
  DISCR 88
  DISCR 89
  DISCR 90
  DISCR 91
  DISCR 92
  DISCR 93

```

## Statistics - Factor Analysis

Factor analysis is a method of analyzing the inter-correlations 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. VARMAX - 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  $\lambda_1, \lambda_2, \dots, \lambda_m$  must be arranged in descending order.

Cumulative percentage for these  $k$  eigenvalues are:

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

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

$m$  = 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 VARMAX 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)
DIMENSION R(11),D(11)
FM=M
L=0
DO 100 I=1,M
  L=L+1
100 D(I)=R(L)
K=0
C      TEST WHETHER I-TH EIGENVALUE IS GREATER
C      THAN OR EQUAL TO THE CONSTANT
DO 110 I=1,M
  IF(D(I)-CON) 120, 105, 105
105 K=K+1
110 D(I)=D(I)/FM
C      COMPUTE CUMULATIVE PERCENTAGE OF EIGENVALUES
120 DO 130 I=2,K
130 D(I)=D(I)+D(I-1)
RETURN
END
```

```
TRACE 1
TRACE 2
TRACE 3
TRACE 4
TRACE 5
TRACE 6
TRACE 7
TRACE 8
TRACE 9
TRACE 10
TRACE 11
TRACE 12
TRACE 13
TRACE 14
TRACE 15
TRACE 16
TRACE 17
TRACE 18
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.

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

where  $i = 1, 2, \dots, m$  are variables

$j = 1, 2, \dots, k$  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 (N,K,R,V)
  DIMENSION R(1),V(1)
  L=0
  JJ=0
  DO 160 J=1,K
    JJ=JJ+J
    150 SQ=SQRT(R(JJ))
    DO 160 I=1,M
      L=L+1
      160 V(L)=SQ*V(I)
    RETURN
  END

```

```

LOAD 1
LOAD 2
LOAD 3
LOAD 4
LOAD 5
LOAD 6
LOAD 7
LOAD 8
LOAD 9
LOAD 10
LOAD 11
LOAD 12

```

# VARMX

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

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

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

Communalities:

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

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

Normalized factor matrix:

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

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

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

Variance for factor matrix:

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

where  $c = 1, 2, \dots$  (iteration cycle)

Convergence test:

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

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  $\phi$ , the angle of rotation. The computational steps are as follows:

#### A. Calculation of NUM and DEN:

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

$$B = 2 \sum_i x_i y_i$$

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

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

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

$$\text{NUM} = D - 2AB/m$$

$$\text{DEN} = C - [(A + B)(A - B)] / m$$

#### 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.

\*  $\epsilon$  is a small tolerance factor.

$$\text{B1: } \tan 4\theta = |\text{NUM}| / |\text{DEN}| \quad (8)$$

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.

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

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

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

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

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

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

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

$\sin \phi = \sin \theta$  (19)  
and go to (D2) below.

If DEN is negative, calculate

$$\cos \phi = \frac{\sqrt{2}}{2} \cos \theta + \frac{\sqrt{2}}{2} \sin \theta \quad (20)$$

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

and go to (D2) below.

D2: If NUM is positive, set

$$\cos \phi = |\cos \phi| \quad (22)$$

$$\sin \phi = |\sin \phi| \quad (23)$$

and go to (E) below.

If NUM is negative, set

$$\cos \phi = -|\cos \phi| \quad (24)$$

$$\sin \phi = -|\sin \phi| \quad (25)$$

E. Rotation:

$$X_i = x_i \cos \phi + y_i \sin \phi \quad (26)$$

$$Y_i = x_i \sin \phi + y_i \cos \phi \quad (27)$$

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

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:

$$a_{ij} = b_{ij} \cdot h_i \quad (28)$$

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

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

Check on communalities:

$$\text{Final communalities } f_i^2 = \sum_{j=1}^k a_{ij}^2 \quad (29)$$

$$\text{Difference } d_i = h_i^2 - f_i^2 \quad (30)$$

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

## Subroutine VARMAX

Purpose:

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

Usage:

CALL VARMAX (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,M,F,D)
DIMENSION A(1),TV(1),H(1),F(1),D(1)
C INITIALIZATION
EPS=0.0016
TVLT=0.0
LL=M-1
NV=1
NC=0
FN=M
FFN=FN*FN
C CALCULATE ORIGINAL COMMUNITIES
DO 110 I=1,M
H(I)=0.0
DO 110 J=1,K
L=M*(J-1)+1
110 H(I)=H(I)+A(L)*A(L)
C CALCULATE NORMALIZED FACTOR MATRIX
DO 120 I=1,M
115 H(I)=SQRT(H(I))
DO 120 J=1,K
L=M*(J-1)+1
120 A(L)=A(L)/H(I)
GO TO 132
C CALCULATE VARIANCE FOR FACTOR MATRIX
130 NV=NV+1
TVLT=TVINV-1
132 TVINV=0.0
DO 150 J=1,K
AA=0.0
BB=0.0
LH=M*(J-1)
DO 140 I=1,M
L=L+1
CC=A(L)*A(L)
AA=AA+CC
140 BB=BB+CC*CC
150 TVINV=TVINV+(FV*BB-AA*AA)/FFN
IF(NV=5) 160, 430, 430
C PERFORM CONVERGENCE TEST
160 IF((TVINV-TVLT)-(1.E-7)) 170, 170, 190
170 NC=NC+1
IF(NC=3) 190, 190, 430
C ROTATION OF TWO FACTORS CONTINUES UP TO
THE STATEMENT 120.
190 DO 420 J=1,LL
LI=M*(J-1)
II=J+1
C CALCULATE NUM AND DEN
DO 420 K=1,K
L2=M*(K-1)
AA=0.0
BB=0.0
CC=0.0
DD=0.0
DO 230 I=1,M
L3=L+1
L4=L+1
U=A(L3)*A(L4)+A(L3)*A(L4)
T=A(L3)*A(L4)
T=T+T
CC=CC+U*U*(U-T)
DD=DD+2.0*U*T
230 AA=AA+U
T=DD-2.0*AA*BB/FN
B=CC-1.0*AA*AA-BB*BB/FN
C COMPARISON OF NUM AND DEN
IF(T-B) 280, 240, 320
240 IF((T+B)-EPS) 420, 250, 250
NUM = DEN IS GREATER THAN OR EQUAL TO THE
TOLERANCE FACTOR
250 COS4T=CONS
SIN4T=CONS
GO TO 350
C NUM IS LESS THAN DEN
280 TAN4T=ABS(T)/ABS(B)
IF(TAN4T=EPS) 300, 290, 290
290 COS4T=1.0/SQRT(1.0+TAN4T*TAN4T)
SIN4T=TAN4T/COS4T
GO TO 360
300 IF(B) 310, 420, 420
310 SINP=CONS
COSPP=CONS
GO TO 400
C NUM IS GREATER THAN DEN
320 CTN4T=ABS(T/B)
IF(CTN4T=EPS) 340, 330, 330
330 SIN4T=1.0/SQRT(1.0+CTN4T*CTN4T)
COS4T=CTN4T/SIN4T
GO TO 350
340 COS4T=0.0
SIN4T=1.0
C DETERMINE COS THETA AND SIN THETA
350 COS2T=SQRT(1.0+COS4T/2.0)
SIN2T=SIN4T/(2.0*COS2T)
355 COST=SQRT(1.0+OS2T)/2.0
SINT=SIN2T/(2.0*COST)
C DETERMINE COS PHI AND SIN PHI
IF(R) 370, 370, 360
360 COSP=COST
SINP=SINT
GO TO 380
370 COSP=CUNS*COST+CJNS*SINT
375 SINP=ABS(CJNS*COST-CUNS*SINT)
380 IF(1) 390, 390, 400
390 SINP=-SINP
C PERFORM ROTATION
400 DO 410 I=1,M
L3=L+1
L4=L+1
AA=A(L3)*COSPP+A(L4)*SINP
A(L4)=-A(L3)*SINP+A(L4)*COSP
410 A(L3)=AA
420 CONTINUE
GO TO 130
C DENDORMALIZE VARIMAX LOADINGS
430 DO 440 I=1,M
DO 440 J=1,K
L=M*(J-1)+1
440 A(L)=A(L)*H(I)
C CHECK ON COMMUNITIES
NC=NV-1
DO 450 I=1,M
H(I)=H(I)+H(I)
450 H(I)=H(I)+H(I)
DO 470 I=1,M
F(I)=0.0
DO 460 J=1,K
L=M*(J-1)+1
460 F(I)=F(I)+A(L)*A(L)
470 D(I)=H(I)-F(I)
RETURN
END

```

```

VARMX 1
VARMX 2
VARMX 3
VARMX 4
VARMX 5
VARMX 6
VARMX 7
VARMX 8
VARMX 9
VARMX 10
VARMX 11
VARMX 12
VARMX 13
VARMX 14
VARMX 15
VARMX 16
VARMX 17
VARMX 18
VARMX 19
VARMX 20
VARMX 21
VARMX 22
VARMX 23
VARMX 24
VARMX 25
VARMX 26
VARMX 27
VARMX 28
VARMX 29
VARMX 30
VARMX 31
VARMX 32
VARMX 33
VARMX 34
VARMX 35
VARMX 36
VARMX 37
VARMX 38
VARMX 39
VARMX 40
VARMX 41
VARMX 42
VARMX 43
VARMX 44
VARMX 45
VARMX 46
VARMX 47
VARMX 48
VARMX 49
VARMX 50
VARMX 51
VARMX 52
VARMX 53
VARMX 54
VARMX 55
VARMX 56
VARMX 57
VARMX 58
VARMX 59
VARMX 60
VARMX 61
VARMX 62
VARMX 63
VARMX 64
VARMX 65
VARMX 66
VARMX 67
VARMX 68
VARMX 69
VARMX 70
VARMX 71
VARMX 72
VARMX 73
VARMX 74
VARMX 75
VARMX 76
VARMX 77
VARMX 78
VARMX 79
VARMX 80
VARMX 81
VARMX 82
VARMX 83
VARMX 84
VARMX 85
VARMX 86
VARMX 87
VARMX 88
VARMX 89
VARMX 90
VARMX 91
VARMX 92
VARMX 93
VARMX 94
VARMX 95
VARMX 96
VARMX 97
VARMX 98
VARMX 99
VARMX100
VARMX101
VARMX102
VARMX103
VARMX104
VARMX105
VARMX106
VARMX107
VARMX108
VARMX109
VARMX110
VARMX111
VARMX112
VARMX113
VARMX114
VARMX115
VARMX116
VARMX117
VARMX118
VARMX119
VARMX120
VARMX121
VARMX122
VARMX123
VARMX124
VARMX125
VARMX126
VARMX127
VARMX128
VARMX129
VARMX130
VARMX131
VARMX132
VARMX133

```

## Statistics - Time Series

### AUTO

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

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

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

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)
DIMENSION A(1),R(1)
C CALCULATE AVERAGE OF TIME SERIES A
AVER=0.0
IF(N=L) 50,50,100
50 R(1)=0.0
RETURN
100 DO 110 I=1,N
110 AVER=AVER+A(I)
PN=N
AVER=AVER/PN
C CALCULATE AUTOCOVARIANCES
DO 130 J=1,L
NJ=N-J+1
SUM=0.0
DO 120 I=1+NJ
IJ=I+J-1
120 SUM=SUM+(A(I)-AVER)*(A(IJ)-AVER)
PNJ=NJ
130 R(J)=SUM/PNJ
RETURN
END

```

```

AUTO 1
AUTO 2
AUTO 3
AUTO 4
AUTO M01
AUTO M02
AUTO M03
AUTO M04
AUTO 6
AUTO 7
AUTO 8
AUTO 11
AUTO 12
AUTO 13
AUTO 14
AUTO 15
AUTO 16
AUTO M05
AUTO 18
AUTO M06
AUTO 20
AUTO 21

```

## CROSS

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

(a) B lags A:

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

(b) B leads A:

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

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

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

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)
  DIMENSION A(1),B(1),R(1),S(1)
  C  CALCULATE AVERAGES OF SERIES A AND B
  FN=N
  AVERA=0.0
  AVERB=0.0
  IF(N=L)50,50,100
50  R(1)=0.0
  S(1)=0.0
  RETURN
100 DO 110 I=1,N
  AVFRA=AVERA+A(I)
110  AVERB=AVERB+B(I)
  AVERA=AVFRA/FN
  AVERB=AVERB/FN
  C  CALCULATE CROSSCOVARIANCES OF SERIES A AND B
  DO 130 J=1,L
  SUMR=0.0
  SUMS=0.0
  DO 120 I=1,NJ
  IJ=I+J-1
  SUMR=SUMR+(A(I)-AVERA)*(B(IJ)-AVERB)
120  SUMS=SUMS+(A(IJ)-AVFRA)*(B(I)-AVERB)
  R(IJ)=SUMR/FN
  S(IJ)=SUMS/FN
130  RETURN
END

```

```

CROSS 1
CROSS 2
CROSS 3
CROSS 4
CROSS 5
CROSS 6
CROSSM01
CROSSM02
CROSSM03
CROSSM04
CROSSM05
CROSS 8
CROSS 9
CROSS 10
CROSS 11
CROSS 14
CROSS 15
CROSS 16
CROSS 17
CROSS 18
CROSS 19
CROSS 20
CROSSM06
CROSSM07
CROSS 23
CROSSM08
CROSSM09
CROSS 26
CROSS 27

```

**SMO**

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

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

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

$$k = i - IL + 1$$

$$i = IL \text{ to } IH$$

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

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

L = a given selection integer. For example, L = 4 applies weights to every 4<sup>th</sup> 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<sup>th</sup> 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

$$IL = (L*(M-1))/2 + 1 \quad \text{..... (1)}$$

$$IH = N - (L*(M-1))/2 \quad \text{..... (2)}$$

#### 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,M,L,R)
  DIMENSION A(1),M(1),R(1)
  C      INITIALIZATION
  DO 110 I=1,N
110 R(I)=0.0
  IL=(L*(M-1))/2+1
  IH=N-(L*(M-1))/2
  C      SMOOTH SERIES A BY WEIGHTS M
  DO 120 J=IL,IH
    R(J)=0.0
  DO 120 J=1,M
    IP=(J*IL)-L+K
120 R(I)+R(I)+R(IP)*W(I)
  RETURN
END

```

```

SMO 1
SMO 2
SMO 3
SMO 4
SMO 5
SMO 6
SMO 7
SMO 8
SMO 9
SMO 10
SMO 11
SMO 12
SMO 13
SMO 14
SMO 15

```

#### EXSMO

This subroutine calculates a smoothed series  $S_1, S_2, \dots, S_{NX}$ , given time series  $X_1, X_2, \dots, X_{NX}$  and a smoothing constant  $\alpha$ . 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  $i = 1, 2, \dots, NX$ , 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

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

- (b) Update coefficients A, B, and C

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

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

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

where  $\alpha$  = 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:

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

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

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

#### 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  

$$A + B \cdot T + C \cdot T \cdot T / 2.$$
  
 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)
  DIMENSION X(1),S(1)
  C IF A=B=C=0.0, GENERATE INITIAL VALUES OF A, B, AND C
  IF(A) 140, 110, 140
110 IF(B) 140, 120, 140
120 IF(C) 140, 130, 140
130 L1=1
  L2=2
  L3=3
  C=X(L1)-2.0*X(L2)+X(L3)
  B=X(L2)-X(L1)-1.5*C
  A=X(L1)-B+0.5*C
140 BECUB=BE*BE*BE
  ALCUB=AL*AL*AL
  DO 150 I=1,NX
  C DO THE FOLLOWING FOR I=1 TO NX
  C FIND S(I) FOR ONE PERIOD AHEAD
  S(I)=A+B*I+0.5*C
  C UPDATE COEFFICIENTS A, B, AND C
  DIF=S(I)-X(I)
  A=X(I)+BECUB*DIF
  B=B+C-1.5*AL*AL*(2.0-AL)*DIF
150 C=C-ALCUB*DIF
  RETURN
  END

```

## 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:

$$d.f. = (n - 1) (m - 1) \quad (1)$$

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:

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

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

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

Chi-square is obtained for two cases:

(a) for 2 x 2 table:

$$\chi^2 = \frac{GT \left( \left| \begin{matrix} A_{11} & A_{22} \\ A_{12} & A_{21} \end{matrix} \right| - \frac{GT^2}{2} \right)^2}{(A_{11} + A_{12})(A_{21} + A_{22})(A_{11} + A_{21})(A_{12} + A_{22})} \quad (5)$$

(b) for other contingency tables:

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

$$\text{where } E_{ij} = \frac{T_i T_j}{GT}$$

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

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

## 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.

```
C      COMPUTE CHI SQUARE FOR OTHER CONTINGENCY TABLES
130  IJ=0
      DO 140 J=1,M
      DO 140 I=1,N
      IJ=IJ+1
      E=TR(I)*TC(J)/GT
      IF(I=1.0) 135, 140, 140
135  IERR=1
140  CS=CS+(A(I,J)-E)*A(I,J)-E)/E
      RETURN
      END
```

CHISQ 48  
CHISQ 49  
CHISQ 50  
CHISQ 51  
CHISQ 52  
CHISQ 53  
CHISQM02  
CHISQM03  
CHISQ 54  
CHISQ 55  
CHISQ 56

```
SUBROUTINE CHISQ(A,N,M,CS,NDF,IERR,TR,TC)
      DIMENSION A(1),TR(1),TC(1)
      NM=N*M
      IERR=0
      CS=0.0
C      FIND DEGREES OF FREEDOM
      NDF=(N-1)*(M-1)
      IF(NDF) 5,5,10
5      IERR=3
      RETURN
C      COMPUTE TOTALS OF ROWS
10  DO 90 I=1,N
      TR(I)=0.0
      IJ=1
      DO 90 J=1,M
      IJ=IJ+1
90  TR(I)=TR(I)+A(IJ)
C      COMPUTE TOTALS OF COLUMNS
      IJ=0
      DO 100 J=1,M
      TC(J)=0.0
      DO 100 I=1,N
      IJ=IJ+1
100  TC(J)=TC(J)+A(IJ)
C      COMPUTE GRAND TOTAL
      GT=0.0
      DO 110 I=1,N
      GT=GT+TR(I)
C      COMPUTE CHI SQUARE FOR 2 BY 2 TABLE (SPECIAL CASE)
120  IF(NM=4) 130,120,130
      L1=1
      L2=2
      L3=3
      L4=4
      CS=GT*(ABS(A(L1)*A(L4)-A(L2)*A(L3))-GT/2.0)**2/(TC(L1)*TC(L2)
      *TR(L1)*TR(L2))
      RETURN
```

CHISQ 1  
CHISQ 2  
CHISQ 3  
CHISQ 4  
CHISQ 5  
CHISQ 6  
CHISQ 7  
CHISQ 8  
CHISQ 9  
CHISQ 10  
CHISQ 25  
CHISQM01  
CHISQ 27  
CHISQ 28  
CHISQ 29  
CHISQ 30  
CHISQ 31  
CHISQ 32  
CHISQ 33  
CHISQ 34  
CHISQ 35  
CHISQ 36  
CHISQ 37  
CHISQ 38  
CHISQ 39  
CHISQ 40  
CHISQ 41  
CHISQ 42  
CHISQ 43  
CHISQ 44  
CHISQM04  
CHISQM05  
CHISQM06  
CHISQM07  
CHISQM08  
CHISQM09  
CHISQ 47

## 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:

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

where  $n_1$  = number of cases in smaller group

$n_2$  = number of cases in larger group

$$U = n_1 n_2 - U'$$

if  $U' < U$ , set  $U = U'$  (2)

A correction factor for ties is obtained:

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

where  $t$  = number of observations tied for a given rank

The standard deviation is computed for two cases:

(a) if  $T = 0$

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

(b) if  $T > 0$

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

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

The significance of  $U$  is then tested:

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

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

$Z$  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)
DIMENSION A(1),R(1)
C   RANK SCORES FROM BOTH GROUP TOGETHER IN ASCENDING ORDER, AND
C   ASSIGN TIED OBSERVATIONS AVERAGE OF TIED RANKS
N=N1+N2
CALL RANK(A,R,N)
Z=0.0
C   SUM RANKS IN LARGER GROUP
R2=0.0
NP=N1+1
DO 10 I=NP,N
R2=R2+R(I)
10 R2=R2+R(I)
C   CALCULATE U
FNX=N1*N2
FN=N
FN2=N2
UP=FNX+FN2*((FN2+1.0)/2.0)-R2
UP=FNX-UP
IF (UP<0) 20,30,33
20 U=UP
C   TEST FOR N2 LESS THAN 20
30 IF (N2<20) 80,40,40
C   COMPUTE STANDARD DEVIATION
40 KT=1
CALL TIE(R,N,KT,TS)
IF (TS) 50,60,50
50 S=SQRT((FNX*(FN*(FN-1.0)))/((FN*FN*FN-1.0)/2.0)-TS))
GO TO 70
60 S=SQRT(FNX*(FN+1.0)/12.0)
C   COMPUTE Z
70 Z=(U-FNX*0.5)/S
80 RETURN
END
UTEST 1
UTEST 2
UTEST 3
UTEST 4
UTEST 5
UTEST 6
UTEST 7
UTEST 8
UTEST 9
UTEST 10
UTEST 11
UTEST 12
UTEST 13
UTEST 14
UTEST 15
UTEST 16
UTEST 17
UTEST 18
UTEST 19
UTEST 20
UTEST 21
UTEST 22
UTEST 23
UTEST 24
UTEST 25
UTEST 26
UTEST 27
UTEST 28
UTEST 29
UTEST 30
UTEST 31
UTEST 32
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:

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

Friedman's statistic is then computed:

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

The degrees of freedom are:

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

## 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)
DIMENSION A(1),R(1),W(1)
      DETERMINE WHETHER DATA IS RANKED
      IF (NR-1) 10, 30, 10
      RANK DATA IN EACH GROUP AND ASSIGN TIED OBSERVATIONS AVERAGE
      OF TIED RANK
10 DO 20 I=1,N
   IJ=I-N
   IK=I-J
   DO 15 J=1,M
     IJ=I+J
     W(IJ)=A(IJ)
     CALL RANK (W,WIM+1),M)
     DO 20 J=1,M
       IK=IK+N
       W(IK)=W(IK)
20 R(IK)=W(IK)
   GO TO 15
30 NM=NM+1
   DO 12 I=1,NM
     32 R(I)=A(I)
      CALCULATE SUM OF SQUARES (IF SUMS OF RANKS
35 RTSQ=0.0
      IR=0
      DO 50 J=1,M
        RT=0.0
        DO 40 I=1,N
          IR=IR+1
40 RT=RT+R(I)
50 RTSQ=RTSQ+RT*RT
      CALCULATE FRIEDMAN TEST VALUE, XR
      FNM=N*(M+1)
      FNM=N*(M+1)
      XR=(12.0/(FNM*N)) * RTSQ - 3.0 * FNM
      FIND DEGREES OF FREEDOM
      NDF=N-1
      RETURN
END

```

## 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:

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

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

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

where  $j = 1, 2, \dots, m$

The Cochran Q statistic is computed:

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

The degrees of freedom are:

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

## 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)
DIMENSION A(1)
C COMPUTE SUM OF SQUARES OF ROW TOTALS, RSQ, AND GRAND TOTAL OF
C ALL ELEMENTS, GD
RSQ=0.0
GD=0.0
DO 20 I=1,N
TR=0.0
IJ=I-N
DO 10 J=1,M
IJ=IJ+N
10 TR=TR+A(IJ)
GD=GD+TR
20 RSQ=RSQ+TR*TR
C COMPUTE SUM OF SQUARES OF COLUMN TOTALS, CSQ
CSQ=0.0
IJ=0
DO 40 J=1,M
TC=0.0
DO 30 I=1,N
IJ=IJ+1
30 TC=TC+A(IJ)
40 CSQ=CSQ+TC*TC
C COMPUTE COCHRAN Q TEST VALUE
FM=M
Q=(FM-1.0)*(FM*CSQ-GD*GD)/(FM*GD-RSQ)
C FIND DEGREES OF FREEDOM
NDF=M-1
RETURN
END
```

## 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:

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

where  $A_i$  = first ranked vector

$B_i$  = second ranked vector

$n$  = number of ranks

A correction factor for ties is obtained:

$$T_a = \sum \frac{t^3 - t}{12} \text{ over variable A} \quad (2)$$
$$T_b = \sum \frac{t^3 - t}{12} \text{ over variable B}$$

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,

$$r_s = 1 - \frac{6D}{n^3 - n} \quad (3)$$

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

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

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

$$Y = \frac{n^3 - n}{12} - T_b \quad (6)$$

The statistic used to measure the significance of  $r_s$  is:

$$t = r_s \sqrt{\frac{n-2}{1-r_s^2}} \quad (7)$$

The degrees of freedom are:

$$\text{d.f.} = n - 2 \quad (8)$$

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

```

      CALL RANK (B,R(N+1),N)
      GO TO 40
C     MOVE RANKED DATA TO R VECTOR
10 DO 20 I=1,N
20 R(I)=A(I)
   DO 30 J=1,N
   J=J+1
30 R(J)=B(I)
C     COMPUTE SUM OF SQUARES OF RANK DIFFERENCES
40 D=0.0
   DO 50 I=1,N
   J=I+N
50 D=D+(R(I)-R(J))*(R(I)-R(J))
C     COMPUTE TIED SCORE INDEX
      KT=1
      CALL TIE (R,N+KT,TSA)
      CALL TIE (R,N+1),N+KT,TSB)
C     COMPUTE SPEARMAN RANK CORRELATION COEFFICIENT
      IF(TSA) 60,55,60
55 IF(TSB) 60,57,60
57 RS=1.0-6.0*D/FNNN
      GO TO 70
60 X=FNNN/12.0-TSA
   Y=X+TSB-TSB
   RS=(X+Y-D)/(2.0*(SORT(X*Y)))
C     COMPUTE T AND DEGREES OF FREEDOM IF N IS 10 OR LARGER
      T=0.0
70 IF(N=10) 80,75,75
75 T=RS*SQRT(FLOAT(N-2)/(1.0-RS*RS))
80 NDF=N-2
      RETURN
      END

```

```

SRANK 9
SRANK 10
SRANK 11
SRANK 12
SRANK 13
SRANK 14
SRANK 15
SRANK 16
SRANK 17
SRANK 18
SRANK 19
SRANK 20
SRANK 21
SRANK 22
SRANK 23
SRANK 24
SRANK 25
SRANK 26
SRANK 27
SRANK 28
SRANK 29
SRANK 30
SRANK 31
SRANK 32
SRANK 33
SRANK 34
SRANK 35
SRANK 36
SRANK 37
SRANK 38
SRANK 39
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:

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

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

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,

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

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,

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

The standard deviation is calculated:

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

The significance of  $\tau$  can be measured by:

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

## 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)
DIMENSION A(1),B(1),R(1)
SD=0.0
Z=0.0
PN=N
FNI=N*(N-1)
C DETERMINE WHETHER DATA IS RANKED
IF(NR-1) 5, 10, 5
C RANK DATA IN A AND B VECTORS AND ASSIGN TIED OBSERVATIONS
C AVERAGE OF TIED RANKS
5 CALL RANK (A,R,N)
CALL RANK (B,R(N),N)
GO TO 40
C MOVE RANKED DATA TO R VECTOR
10 DO 20 I=1,N
20 R(I)=R(I)
DO 30 I=1,N
J=I+N
30 R(J)=R(I)
C SORT RANK VECTOR R IN SEQUENCE OF VARIABLE A
40 ISORT=0
DO 50 I=2,N
IF(R(I)-R(I-1)) 45,50,50
45 ISORT=ISORT+1
RSAVE=R(I)
R(I)=R(I-1)
R(I-1)=RSAVE
I2=I+N
SAVER=R(I2)
R(I2)=R(I2-1)
R(I2-1)=SAVER
50 CONTINUE
IF(ISORT) 40,55,40
C COMPUTE S ON VARIABLE B, STARTING WITH THE FIRST RANK, ADD 1
C TO S FOR EACH LARGER RANK TO ITS RIGHT AND SUBTRACT 1 FOR EACH
C SMALLER RANK. REPEAT FOR ALL RANKS.
```

```
55 S=0.0
NM=N-1
DO 60 I=1,NM
J=N-I
DO 60 L=I,N
K=N+L
IF(R(K)-R(J)) 56,60,57
56 S=S-L.0
GO TO 60
57 S=S+L.0
60 CONTINUE
C COMPUTE TIED SCORE INDEX FOR BOTH VARIABLES
KT=2
CALL TIE(R,N,KT,TA)
CALL TIE(R(N),N,KT,TR)
C COMPUTE TAU
IF(TA) 70,85,70
65 IF(TR) 70,67,70
67 TAU=S/(0.5*FNI)
GO TO 90
C TAU=S/((ISORT(0.5*FNI-TA))+ISORT(0.5*FNI-TR))
70 TAU=S/((ISORT(0.5*FNI-TA))+ISORT(0.5*FNI-TR))
80 IF(N-10) 90,85,85
85 SD=ISORT(2.0*(FNI*FNI+5.01)/(0.5*FNI))
Z=TAU/SD
90 RETURN
END
```

KRANK 37  
KRANK 38  
KRANK 39  
KRANK 40  
KRANK 41  
KRANK 42  
KRANK 43  
KRANK 44  
KRANK 45  
KRANK 46  
KRANK 47  
KRANK 48  
KRANK 49  
KRANK 50  
KRANK 51  
KRANK 52  
KRANK 53  
KRANK 54  
KRANK 55  
KRANK 56  
KRANK 57  
KRANK 58  
KRANK 59  
KRANK 60  
KRANK 61  
KRANK 62

## 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:

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

where t = number of observations tied for a given rank

Sums of ranks are calculated:

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

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

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

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

The sum of squares of deviations is derived:

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

The Kendall coefficient of concordance is then computed:

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

For m larger than 7, chi-square is:

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

The degrees of freedom are:

$$\text{d.f.} = n - 1 \quad (7)$$

## 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)
  DIMENSION A(1),R(1),WA(1)
  FM=M
  FN=N
  C
  DETERMINE WHETHER DATA IS RANKED
  RANK DATA FOR ALL VARIABLES ASSIGNING TIED OBSERVATIONS AVERAGE OF
  OF TIED RANKS AND COMPUTE CORRECTION FOR TIED SCORES
  T=0.0
  KT=1
  DO 20 J=1,M
    IJ=J-1
    IK=IJ
    IF (NR-IJ) 5,2,5
    DO 3 J=1,M
      IJ=J+N
      K=K+J
      3 WA(IK+WA(IJ))
      GO TO 15
    5 DO 10 J=1,M
      IJ=IJ+N
      10 WA(IJ)=WA(IJ)
      CALL RANK(WA,WA(IJ+1),M)
      15 CALL TIE(WA(M+1),M,KT,T)
      T=T+T
      DO 20 J=1,M
        IK=IK+N
        20 R(IK)=WA(IJ)
  C
  CALCULATE VECTOR OF SUMS OF RANKS
  IR=0
  DO 40 J=1,M
    WA(IJ)=0.0
    DO 40 I=1,N
      IR=IR+1
      40 WA(IJ)=WA(IJ)+R(IJ)
  C
  COMPUTE MEAN OF SUMS OF RANKS
  SM=0.0
  DO 50 J=1,M
    50 SM=SM+WA(IJ)
    SM=SM/FM
  C
  COMPUTE SUM OF SQUARES OF DEVIATIONS
  S=0.0
  DO 60 J=1,M
    60 S=S+(WA(IJ)-SM)*(WA(IJ)-SM)
  C
  COMPUTE W
  W=S/((FM*FN)*(FM+FN-FM)/12.0)-FN*T)
  C
  COMPUTE DEGREES OF FREEDOM AND CHI-SQUARE IF M IS OVER 7
  CS=0.0
  NDF=0
  IF (M-1) 70,70,65
  65 CS=FN*(FN-1.0)*W
  NDF=M-1
  70 RETURN
END
WTEST 1
WTEST 2
WTEST 3
WTEST 4
WTEST 5
WTEST 6
WTEST 7
WTEST 8
WTEST 9
WTEST 10
WTEST 11
WTEST 12
WTEST 13
WTEST 14
WTEST 15
WTEST 16
WTEST 17
WTEST 18
WTEST 19
WTEST 20
WTEST 21
WTEST 22
WTEST 23
WTEST 24
WTEST 25
WTEST 26
WTEST 27
WTEST 28
WTEST 29
WTEST 30
WTEST 31
WTEST 32
WTEST 33
WTEST 34
WTEST 35
WTEST 36
WTEST 37
WTEST 38
WTEST 39
WTEST 40
WTEST 41
WTEST 42
WTEST 43
WTEST 44
WTEST 45
WTEST 46
WTEST 47
WTEST 48
WTEST 49
WTEST 50
WTEST 51
WTEST 52
WTEST 53
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)
  DIMENSION A(1),R(1)
  C      INITIALIZATION
  DO 10 I=1,N
  C      R(I)=0.0
  C      FIND RANK OF DATA
  DO 100 J=1,N
  C      TEST WHETHER DATA POINT IS ALREADY RANKED
  IF(R(J)) 20, 20, 100
  C      DATA POINT TO BE RANKED
  20 SMALL=0.0
  EQUAL=0.0
  X=A(I)
  DO 50 J=1,N
  C      IF(A(J)=X) 30, 40, 50
  C      COUNT NUMBER OF DATA POINTS WHICH ARE SMALLER
  30 SMALL=SMALL+1.0
  GO TO 50
  C      COUNT NUMBER OF DATA POINTS WHICH ARE EQUAL
  40 EQUAL=EQUAL+1.0
  R(J)=SMALL+1.0
  50 CONTINUE
  C      TEST FOR TIE
  IF(EQUAL=1.0) 60, 60, 70
  C      STORE RANK OF DATA POINT WHERE NO TIE
  60 R(I)=SMALL+1.0
  GO TO 100
  C      CALCULATE RANK OF TIED DATA POINTS
  70 P=SMALL*(EQUAL+1.0)/2.0
  DO 90 J=1,N
  C      IF(R(J)=X) 90, 80, 90
  80 R(J)=P
  90 CONTINUE
  100 CONTINUE
  RETURN
  END
```

```

RANK 1
RANK 2
RANK 3
RANK 4
RANK 5
RANK 6
RANK 7
RANK 8
RANK 9
RANK 10
RANK 11
RANK 12
RANK 13
RANK 14
RANK 15
RANK 16
RANK 17
RANK 18
RANK 19
RANK 20
RANK 21
RANK 22
RANK 23
RANK 24
RANK 25
RANK 26
RANK 27
RANK 28
RANK 29
RANK 30
RANK 31
RANK 32
RANK 33
RANK 34
RANK 35
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  $T = \text{SUM}(CT**3 - CT)/12$

Equation 2  $T = \text{SUM}(CT*(CT-1)/2)$

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)
  DIMENSION R(1)
  C      INITIALIZATION
  T=0.0
  Y=0.0
  Y=0.0
  I=0
  C      FIND NEXT LARGEST RANK
  DO 10 I=1,N
  C      IF(R(I)=Y) 30, 30, 10
  10 IF(R(I)=Y) 20, 40, 10
  20 X=R(I)
  I=I+1
  30 CONTINUE
  C      IF ALL RANKS HAVE BEEN TESTED, RETURN
  IF(I=1) 40, 40, 40
  40 Y=X
  CT=0.0
  C      COUNT TIES
  DO 60 J=1,N
  C      IF(R(J)=X) 60, 50, 60
  50 CT=CT+1.0
  60 CONTINUE
  C      CALCULATE CORRECTION FACTOR
  IF(KT) 70, 5, 70
  70 IF(KT=1) 75, 80, 75
  75 T=T+CT*(CT-1)/2.0
  GO TO 5
  80 T=T+CT*CT*CT-CT/12.0
  GO TO 5
  90 RETURN
  END
```

```

TIE 1
TIE 2
TIE 3
TIE 4
TIE 5
TIE 6
TIE 7
TIE 8
TIE 9
TIE 10
TIE 11
TIE 12
TIE 13
TIE 14
TIE 15
TIE 16
TIE 17
TIE 18
TIE 19
TIE 20
TIE 21
TIE 22
TIE 23
TIE 24
TIE 25
TIE 26
TIE 27
TIE 28
TIE 29
TIE 30
TIE 31
TIE 32
```

## Statistics - Random Number Generators

### 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)
  IY=IX*899
  IF(IY)5,5,6
5  IY=IY/32767+1
6  YFL=IY
  YFL=YFL/32767.
  RETURN
END
```

```
RANDU 1
RANDU 2
RANDU 3
RANDU 4
RANDU 5
RANDU 6
RANDU 7
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:

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

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 asymptotically 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

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

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)
A=0.0
DO 50 I=1,12
CALL RANDU(IX,IY,V)
IX=IY
50 A=A+V
V=(A-6.01)*S+AM
RETURN
END
```

```
GAUSS 1
GAUSS 2
GAUSS 3
GAUSS 4
GAUSS 5
GAUSS 6
GAUSS 7
GAUSS 8
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)
DIMENSION A(1),L(1),M(1)
SEARCH FOR LARGEST ELEMENT
C
D=1.0
NK=-N
DO 80 K=1,N
NK=MAX(N,L(K))
M(K)=K
K=MAX(K,K)
BIGA=A(K)
DO 20 J=K,N
IZ=MAX(J,I)
DO 20 I=K,N
IJ=IZ+1
10 IF (ABS(BIGA)-ABS(A(IJ))) 15,20,20
15 BIGA=A(IJ)
L(K)=I
M(K)=J
20 CONTINUE
C
INTERCHANGE ROWS
J=L(K)
IF(J=K) 35,35,25
25 K1=K-N
DO 30 I=1,N
KI=KI+N
HOLD=A(K)
JI=KI+K
A(KI)=A(JI)
A(JI)=HOLD
30 A(JI)=HOLD
C
INTERCHANGE COLUMNS
35 I=M(K)
IF(I=K) 45,45,38
38 JP=M(I)-1
DO 40 J=1,N
JK=NK+J
JI=JP+J
HOLD=A(JK)
A(JK)=A(JI)
A(JI)=HOLD
40 A(JI)=HOLD
C
DIVIDE COLUMN BY MINUS PIVOT (VALUE OF PIVOT ELEMENT IS
CONTAINED IN BIGA)
C
45 IF(ABS(BIGA)-1.E-20)46,46,48
48 D=D*D
RETURN
48 DO 55 I=1,N
IF(I=K) 50,55,50
50 IK=NK+I
A(IK)=A(IK)/(-BIGA)
55 CONTINUE
C
REDUCE MATRIX
DO 65 I=1,N
IK=NK+I
MINV 1
MINV 2
MINV 3
MINV 4
MINV 5
MINV 6
MINV 7
MINV 8
MINV 9
MINV 10
MINV 11
MINV 12
MINV 13
MINV 14
MINV 15
MINV 16
MINV 17
MINV 18
MINV 19
MINV 20
MINV 21
MINV 22
MINV 23
MINV 24
MINV 25
MINV 26
MINV 27
MINV 28
MINV 29
MINV 30
MINV 31
MINV 32
MINV 33
MINV 34
MINV 35
MINV 36
MINV 37
MINV 38
MINV 39
MINV 40
MINV 41
MINV 42
MINV 43
MINV 44
MINV 45
MINV 46
MINV 47
MINV 48
MINV 49
MINV 50
MINV 51
MINV 52
MINV 53
```

```

      HOLD=A(IK)
      IJ=I-N
      DO 65 J=1,N
      IJ=I+N
      IF(I=K) 60+65+60
60  IF(I=K) 62+65+62
62  KJ=I-J+K
      A(I,J)=HOLD+A(K,J)+A(I,J)
C      CONTINUE
C      DIVIDE ROW BY PIVOT
      KJ=K-N
      DO 75 J=1,N
      KJ=K+N
      IF(I=K) 70+75+70
70  A(K,J)=A(K,J)/BIGA
75  CONTINUE
C      PRODUCT OF PIVOTS
      D=D*BIGA
C      REPLACE PIVOT BY RECIPROCAL
      A(KK)=1.0/BIGA
80  CONTINUE
C      FINAL ROW AND COLUMN INTERCHANGE
      K=N
100  K=K-1
      IF(K) 150+150+105
105  L=L(K)
      IF(I=K) 120+120+108
108  JQ=N*(K-1)
      JR=N*(I-1)
      DO 110 J=1,N
      JK=JQ+J
      HOLD=A(I,K)
      JI=JR+J
      A(I,K)=A(J,I)
110  A(J,I)=HOLD
120  J=M(K)
      IF(I=K) 100+100+125
125  KI=K-N
      DO 130 I=1,N
      KI=K+N
      HOLD=A(K,I)
      JI=K-K+J
      A(K,I)=A(J,I)
130  A(J,I)=HOLD
      GO TO 100
150  RETURN
      END

```

```

MINV M01
MINV 54
MINV 55
MINV 56
MINV 57
MINV 58
MINV 59
MINV M02
MINV 61
MINV 62
MINV 63
MINV 64
MINV 65
MINV 66
MINV 67
MINV 68
MINV 69
MINV 70
MINV 71
MINV 72
MINV 73
MINV 74
MINV 75
MINV 76
MINV 77
MINV 78
MINV 79
MINV 80
MINV 81
MINV 82
MINV 83
MINV 84
MINV 85
MINV 86
MINV 87
MINV 88
MINV 89
MINV 90
MINV 91
MINV 92
MINV 93
MINV 94
MINV 95
MINV 96
MINV 97
MINV 98
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:

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

$\nu_I$  = initial norm

A = input matrix (symmetric)

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

The final norm is computed:

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

This final norm is set sufficiently small that the requirement that any off-diagonal element  $A_{lm}$  shall be smaller than  $\nu_F$  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:

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

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

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

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

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

$$B = A_{11} \cos \theta - A_{1m} \sin \theta \quad (8)$$

$$C = A_{11} \sin \theta + A_{1m} \cos \theta \quad (9)$$

$$B = R_{11} \cos \theta - R_{1m} \sin \theta \quad (10)$$

$$R_{1m} = R_{11} \sin \theta + R_{1m} \cos \theta \quad (11)$$

$$R_{11} = B \quad (12)$$

$$A_{11} = A_{11} \cos^2 \theta + A_{mm} \sin^2 \theta - 2A_{1m} \sin \theta \cos \theta \quad (13)$$

$$A_{mm} = A_{11} \sin^2 \theta + A_{mm} \cos^2 \theta + 2A_{1m} \sin \theta \cos \theta \quad (14)$$

$$A_{1m} = (A_{11} - A_{mm}) \sin \theta \cos \theta + A_{1m} (\cos^2 \theta - \sin^2 \theta) \quad (15)$$

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)
  DIMENSION A(1),R(1)
  GENKATF IDENTIFY MATRIX
  IF(MV=1) 10,25,13
10  IQ=N
  DO 20 J=1,N
    IQ=IQ+N
    DO 20 I=1,N
      IF(I=J) 15,20,15,20
      R(I,J)=1.0
15  R(I,J)=1.0
20  CONTINUE
C  25  ANORM=0.0
  DO 35 I=1,N
    DO 35 J=1,N
      IF(I=J) 30,35,30
      IA=I+(J-I)/2
      ANORM=ANORM+A(IA)*A(IA)
30  IF(ANORM) 165,165,40
40  ANORM=1.414*SQRT(ANORM)
  ANRM=ANORM*1.0E-6/PLD(1,N)
C  35  CONTINUE
  INITIALIZE INDICATORS AND COMPUTE THRESHOLD, THR
  IND=0
  THR=ANORM
45  THR=THR/PLD(1,N)
50  L=1
55  M=L+1
C  60  MQ=(M-M)/2
  LQ=(L+L-L)/2
  LML=MQ
62  IF(ABS(A(L,M))-T4R) 130,65,65
65  IND=1
  LL=L+LQ
  MM=M+MQ
  X=0.5*(A(LL)-A(MM))
68  Y=-A(LL)/SQRT(A(LL)*A(LL)+X*X)
  IF(L) 70,75,75
70  Y=-Y
75  SINX=Y/SQRT(1.0+(SQRT(1.0+Y*Y)))
  SINX2=SINX*SINX
78  COSX=SQRT(1.0-SINX2)
  COSX2=COSX*COSX
  SINCS=SINX*COSX
  ROTATE L AND M COLUMNS
  ILQ=M*(L-1)
  IMQ=M*(M-1)
  DO 125 I=1,N
    IQ=(I+I-1)/2
    IF(I=L) 80,115,80
    IF(I=M) 85,115,80
80  IM=IM+Q
  GO TO 95
95  IF(I=L) 100,105,105
100  IL=IL+Q
  GO TO 110
105  IL=IL+Q
110  X=A(IL)*COSX-A(IM)*SINX
  A(IL)=A(IL)*SINX+A(IM)*COSX
  A(IM)=X
115  IF(M=1) 120,125,120
120  LR=IL+Q
  X=A(ILR)*COSX-R(IMR)*SINX
  R(IMR)=R(IMR)*SINX+R(ILR)*COSX
  R(ILR)=X
125  CONTINUE
  X=2.0*A(LL)*SINCS
  Y=A(LL)*COSX2+A(MM)*SINX2-X
  A(LL)=SINX2+A(MM)*COSX2+X
  A(MM)=A(LL)-A(MM)*SINCS+A(LL)*COSX2-SINX2
  A(LL)=Y
  A(MM)=X
C  TESTS FOR COMPLETION
C  TEST FOR M = LAST COLUMN
130  IF(M=N) 135,140,135
135  M=M+1
  GO TO 60
C  TEST FOR L = SECOND FROM LAST COLUMN
140  IF(L=N-1) 145,150,145
145  L=L+1
  GO TO 55
150  IF(IND=1) 160,155,160
155  IND=0
  GO TO 50
C  160  IF(THR-ANRM) 165,165,65
  SORT EIGENVALUES AND EIGENVECTORS
C  165  IQ=N
  DO 185 I=1,N
    IQ=IQ+N
    LL=I*(I-1)/2
    JQ=N*(I-2)
    DO 185 J=1,N
      JQ=JQ+N
      MM=J*(J-I)/2
      IF(A(LL)-A(MM)) 170,185,185
170  X=A(LL)
  A(LL)=A(MM)
  A(MM)=X
  IF(MV=1) 175,185,175
175  DO 180 K=1,N
    IL=IQ+K
    IM=JQ+K
    X=R(ILR)
    R(ILR)=R(IMR)
180  R(IMR)=X
185  CONTINUE
  RETURN
  END
EIGEN 1
EIGEN 2
EIGEN 3
EIGEN 4
EIGEN 5
EIGEN 6
EIGEN 7
EIGEN 8
EIGEN 9
EIGEN 10
EIGEN 11
EIGEN 12
EIGEN 13
EIGEN 14
EIGEN 15
EIGEN 16
EIGEN 17
EIGEN 18
EIGEN 19
EIGEN 20
EIGEN 21
EIGEN 22
EIGEN 23
EIGEN 24
EIGEN 25
EIGEN 26
EIGEN 27
EIGEN 28
EIGEN 29
EIGEN 30
EIGEN 31
EIGEN 32
EIGEN 33
EIGEN 34
EIGEN 35
EIGEN 36
EIGEN 37
EIGEN 38
EIGEN 39
EIGEN 40
EIGEN 41
EIGEN 42
EIGEN 43
EIGEN 44
EIGEN 45
EIGEN 46
EIGEN 47
EIGEN 48
EIGEN 49
EIGEN 50
EIGEN 51
EIGEN 52
EIGEN 53
EIGEN 54
EIGEN 55
EIGEN 56
EIGEN 57
EIGEN 58
EIGEN 59
EIGEN 60
EIGEN 61
EIGEN 62
EIGEN 63
EIGEN 64
EIGEN 65
EIGEN 66
EIGEN 67
EIGEN 68
EIGEN 69
EIGEN 70
EIGEN 71
EIGEN 72
EIGEN 73
EIGEN 74
EIGEN 75
EIGEN 76
EIGEN 77
EIGEN 78
EIGEN 79
EIGEN 80
EIGEN 81
EIGEN 82
EIGEN 83
EIGEN 84
EIGEN 85
EIGEN 86
EIGEN 87
EIGEN 88
EIGEN 89
EIGEN 90
EIGEN 91
EIGEN 92
EIGEN 93
EIGEN 94
EIGEN 95
EIGEN 96
EIGEN 97
EIGEN 98
EIGEN 99
EIGEN100
EIGEN101
EIGEN102
EIGEN103
EIGEN104
EIGEN105
EIGEN106
EIGEN107
EIGEN108
EIGEN109
EIGEN110
EIGEN111
EIGEN112
EIGEN113
EIGEN114

```

## Mathematics - Matrices

### 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)
  DIMENSION A(1:N,1:M),B(1:N,1:M)
  C
  CALCULATE NUMBER OF ELEMENTS
  NN=NM
  ADD MATRICES
  DO 10 I=1,NN
  10 R(I)=A(I)+B(I)
  RETURN
END
```

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

```
GMSUB 1
GMSUB 2
GMSUB 3
GMSUB 4
GMSUB 5
GMSUB 6
GMSUB 7
GMSUB 8
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)
DIMENSION A(1),B(1),R(1)
IR=0
IK=-M
DO 10 K=1,L
IK=IK+M
DO 10 J=1,N
IR=IR+1
JI=J-N
R(IR)=0
DO 10 I=1,M
JI=JI+N
IR=IR+1
10 R(IR)=R(IR)+A(JI)*B(IR)
RETURN
END
```

```
GMPRD 1
GMPRD 2
GMPRD 3
GMPRD 4
GMPRD 5
GMPRD 6
GMPRD 7
GMPRD 8
GMPRD 9
GMPRD 10
GMPRD 11
GMPRD 12
GMPRD 13
GMPRD 14
GMPRD 15
GMPRD 16
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)
DIMENSION A(1),R(1)
IR=0
DO 10 I=1,N
IJ=I-N
DO 10 J=1,M
IJ=IJ+M
IR=IR+1
10 R(IR)=A(IJ)
RETURN
END
```

```
GMTRA 1
GMTRA 2
GMTRA 3
GMTRA 4
GMTRA 5
GMTRA 6
GMTRA 7
GMTRA 8
GMTRA 9
GMTRA 10
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)
  DIMENSION A(1),B(1),R(1)
  IR=0
  IK=N
  DO 10 K=1,L
    IJ=0
    IK=IK+N
    DO 10 J=1,M
      IR=IR+1
      R(IR)=0
      DO 10 I=1,N
        IJ=IJ+1
        IR=IR+1
      10 R(IR)=R(IR)+A(IJ)*B(1B)
    RETURN
  END
```

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

## 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 MPAD(A,B,R,N,M,MSA,MSB,L)
  DIMENSION A(1:N,1:M),B(1:M,1:L),R(1:N,1:L)
  C SPECIAL CASE FOR DIAGONAL BY DIAGONAL
  IF(MS=22) 30,10,30
  10 DO 20 I=1,N
  20 R(I)=A(I)*B(I)
  RETURN
  C ALL OTHER CASES
  30 IA=1
  DO 90 K=1,L
  DO 90 J=1,M
  R(IJR)=0
  DO 80 I=1,N
  IF(MS) 40,60,40
  40 CALL LOC(I,J,IA,N,M,MSA)
  CALL LOC(I,J,IB,M,L,MSB)
  IF(IA) 50,80,50
  50 IF(IB) 70,80,70
  60 IA=NR(I)-1+J
  IB=NR(K)-1+I
  70 R(IJR)=R(IJR)+A(IA)*B(IB)
  80 CONTINUE
  90 IX=IR+1
  RETURN
END
```

## 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)
  DIMENSION A(1),R(1)
  IF MS IS 1 OR 2, COPY A
  IF(MS) 10,20,10
  10 CALL MCPY(A,R,N,M,MS)
  RETURN
  C TRANSPOSE GENERAL MATRIX
  20 IR=0
  DO 30 I=1,N
    IJ=1-M
    DO 30 J=1,M
      IJ=IJ+N
      IR=IR+1
      R(IJ)=A(IJ)
    RETURN
  END
```

```

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

```

TPRD 1
TPRD 2
TPRD 3
TPRD 4
TPRD 5
TPRD 6
TPRD 7
TPRD 8
TPRD 9
TPRD 10
TPRD 11
TPRD 12
TPRD 13
TPRD 14
TPRD 15
TPRD 16
TPRD 17
TPRD 18
TPRD 19
TPRD 20
TPRD 21
TPRD 22
TPRD 23
TPRD 24
TPRD 25
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)
  DIMENSION A(1),R(1)
  DO 50 K=1,M
    KA=(K*K-K)/2
    DO 60 J=1,M
      IF(J-K) 10,10,60
10  IR=J+K
   R(IR)=0
   DO 80 I=1,N
     IF(MS) 20,40,20
20  CALL LOC(I,J,IA,N,M,MS)
   CALL LOC(I,K,IR,N,M,MS)
   IF(IA) 30,60,30
30  IF(IR) 50,60,50
40  IA=MS*(J-1)+1
   IR=MS*(K-1)+1
50  R(IR)=R(IR)+A(IA)*A(10)
60  CONTINUE
  RETURN
  END
```

## 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)
  DIMENSION A(1),R(1)
  COMPUTE VECTOR LENGTH, IT
  CALL LOC(N,M,IT,N,M,MS)
  ADD SCALAR
  DO 1 I=1,IT
1  R(I)=A(I)+C
  RETURN
  END
```

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

## 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(IRR)=R(IRR)+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)
DIMENSION A(1), R(1)
IR=N*(ICR-1)
DO 2 I=1, N
  IR=IR+1
  C LOCATE INPUT ELEMENT FOR ANY MATRIX STORAGE MODE
  CALL LOC(ICA, ICA, N, M, MS)
  C TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX
  IF(ICA) 1, 2, 1
  C ADD ELEMENTS
  1 R(IR)=R(IR)+A(ICA)
  2 CONTINUE
RETURN
END
```

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

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

```

SCMA 1
SCMA 2
SCMA 3
SCMA 4
SCMA 5
SCMA 6
SCMA 7
SCMA 8
SCMA 9
SCMA 10
SCMA 11
SCMA 12
SCMA 13
SCMA 14
SCMA 15
SCMA 16
SCMA 17
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)
DIMENSION A(1)
LAJ=LA-N
LBJ=LB-N
DO 3 J=1,M
  C LOCATE ELEMENTS IN BOTH ROWS
  LAJ=LAJ+N
  LBJ=LBJ+N
  C INTERCHANGE ELEMENTS
  SAVE=A(LAJ)
  A(LAJ)=A(LBJ)
  A(LBJ)=SAVE
3 RETURN
END
```

```

RINT 1
RINT 2
RINT 3
RINT 4
RINT 5
RINT 6
RINT 7
RINT 8
RINT 9
RINT 10
RINT 11
RINT 12
RINT 13
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
ILA=N*(LA-1)
CINT 4
ILB=N*(LB-1)
CINT 5
DO 3 I=1,N
CINT 6
ILA=ILA+1
CINT 7
ILB=ILB+1
CINT 8
INTERCHANGE ELEMENTS
CINT 9
SAVE=AILA)
CINT 10
AILA)=AILB)
CINT 11
3 AILB)=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,I,J,M,MS)
RSUM 8
C TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX
RSUM 9
IF(I,J) 2,3,2
RSUM 10
C ACCUMULATE IN OUTPUT VECTOR
RSUM 11
2 R(I)=R(I)+A(I,J)
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)
  DIMENSION A(1),R(1)
  DO 3 J=1,M
    C CLEAR OUTPUT LOCATION
    R(J)=0.0
    DO 3 I=1,N
      C LOCATE ELEMENT FOR ANY MATRIX STORAGE MODE
      CALL LOC(I,J,I,J,N,M,MS)
      C TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX
      IF(I,J) 2,3,2
      C ACCUMULATE IN OUTPUT VECTOR
    2 R(J)=R(J)+A(I,J)
    3 CONTINUE
  RETURN
END
```

```

CSUM 1
CSUM 2
CSUM 3
CSUM 4
CSUM 5
CSUM 6
CSUM 7
CSUM 8
CSUM 9
CSUM 10
CSUM 11
CSUM 12
CSUM 13
CSUM 14
CSUM 15
```

## RTAB

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

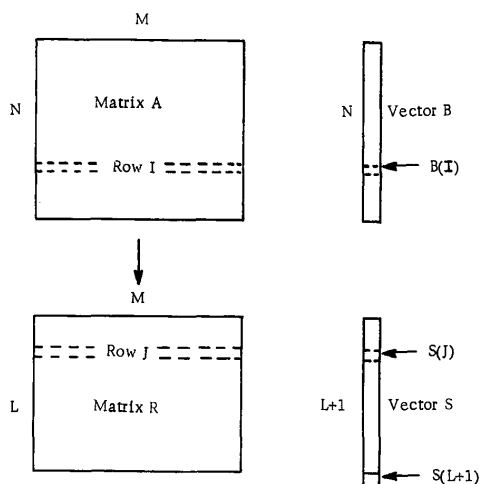


Figure 6. Row tabulation

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

```

## CTAB

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

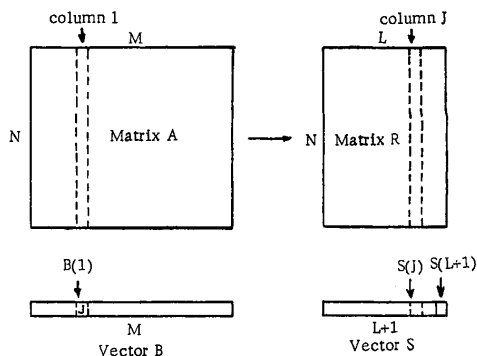


Figure 7. Column tabulation

## 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<sup>th</sup> column of A is added to the J<sup>th</sup> 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)
  DIMENSION A(1),B(1),R(1),S(1)
  CLEAR OUTPUT AREAS
  CALL LOC(N,L,IT,N,L,0)
  DO 10 IR=1,IT
    DO 20 IS=1,L
      S(IS)=0.0
      S(L+1)=0.0
      DO 30 I=1,M
        TEST FOR THE KEY OUTSIDE THE RANGE
        IF(B(I)) 50,50,30
        IF(B(I))-E) 40,40,50
        CALL CADD(A,I,R,J,N,M,MS,L)
        S(JR)=S(JR)+1.0
        GO TO 60
      50 S(L+1)=S(L+1)+1.0
      60 CONTINUE
    RETURN
  END

```

## 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)-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)
  DIMENSION A(1),B(1),R(1)
  C MOVE SORTING KEY VECTOR TO FIRST COLUMN OF OUTPUT MATRIX
  C AND BUILD ORIGINAL SEQUENCE LIST IN SECOND COLUMN
  DO 10 I=1,N
    R(I)=B(I)
    I2=I+N
    10 R(I2)=I
  C SORT ELEMENTS IN SORTING KEY VECTOR (ORIGINAL SEQUENCE LIST
  C IS RESEQUENCED ACCORDINGLY)
  L=N+1
  20 ISORT=0
  L=L-1
  DO 40 I=2,L
    IF(R(I)-R(I-1)) 30,40,40
    30 ISORT=1
    RSAVE=R(I)
    R(I)=R(I-1)
    I2=I+N
    SAVED=R(I2)
    R(I2)=R(I2-1)
    R(I2-1)=SAVED
    40 CONTINUE
  IF(ISORT) 20,50,20
  C MOVE ROWS FROM MATRIX A TO MATRIX R (NUMBER IN SECOND COLUMN
  C OF R REPRESENTS ROW NUMBER OF MATRIX A TO BE MOVED)
  50 DO 80 I=1,N
    I2=I+N
    IN=R(I2)
    IR=I+N
    DO 90 J=1,M
      C LOCATE ELEMENT IN OUTPUT MATRIX
      IR=IR+N
      C LOCATE ELEMENT IN INPUT MATRIX
      CALL LOC(IN,J,IA,N,M,MS)
      C TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX
      IF(IA) 60,70,60
      C MOVE ELEMENT TO OUTPUT MATRIX
      60 R(IR)=A(IA)
      70 R(IR)=0
    80 CONTINUE
  RETURN
  END

```

Mathematics — Matrices 77

## 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)
DIMENSION A(1),B(1),R(1)
C      MOVE SORTING KEY VECTOR TO FIRST ROW OF OUTPUT MATRIX
C      AND BUILD ORIGINAL SEQUENCE LIST IN SECOND ROW
      IK=1
      DO 10 J=1,M
        R(IK)=B(J)
        R(IK+1)=J
      10 IK=IK+N
C      SORT ELEMENTS IN SORTING KEY VECTOR (ORIGINAL SEQUENCE LIST
C      IS RESEQUENCED ACCORDINGLY)
      L=M+1
      20 ISORT=0
      L=L-1
      IP=1
      IQ=N+1
      DO 50 J=2,L
        IF(R(IQ)-R(IP)) 30,40,40
      30 ISORT=1
        RSAVE=R(IQ)
        R(IQ)=R(IP)
        R(IP)=RSAVE
        SAVER=R(IQ+1)
        R(IQ+1)=R(IP+1)
        R(IP+1)=SAVER
      40 IP=IP+N
      IQ=IQ+N
      50 CONTINUE
      IF(ISORT) 20,60,20
C      MOVE COLUMNS FROM MATRIX A TO MATRIX R (NUMBER IN SECOND ROW
C      OF R REPRESENTS COLUMN NUMBER OF MATRIX A TO BE MOVED)
      60 IQ=N
      DO 70 J=1,M
        IQ=IQ+N
C      GET COLUMN NUMBER IN MATRIX A
        IZ=IQ+2
        IN=R(IZ)
C      MOVE COLUMN
        IR=IQ+1
        CALL CCOPY(A,IN,R(IR),N,M,MS)
      70 CONTINUE
      RETURN
      END
CSRT 1
CSRT 2
CSRT 3
CSRT 4
CSRT 5
CSRT 6
CSRT 7
CSRT 8
CSRT 9
CSRT 10
CSRT 11
CSRT M01
CSRT 12
CSRT M02
CSRT 13
CSRT 14
CSRT M03
CSRT 16
CSRT M04
CSRT 18
CSRT 19
CSRT 20
CSRT 21
CSRT 22
CSRT 23
CSRT 24
CSRT 25
CSRT 26
CSRT 27
CSRT 28
CSRT 29
CSRT 30
CSRT 31
CSRT 32
CSRT 33
CSRT 34
CSRT 35
CSRT 36
CSRT 37
CSRT 38
CSRT 39
CSRT 40
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)
  DIMENSION A(1),R(1),S(1)
  IR=0
  IS=0
  DO TO J=1,M
    DO TO I=1,N
      C FIND LOCATION IN OUTPUT MATRIX AND SET TO ZERO
      IF(I-L) 20,10,10
      10 IS=IS+1
      S(1S)=0.0
      GO TO 30
      20 IR=IR+1
      R(1R)=0.0
      C LOCATE ELEMENT FOR ANY MATRIX STORAGE MODE
      30 CALL LOC(I,J,I,J,N,M,MS)
      C TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX
      IF(I,J) 40,70,40
      C DETERMINE WHETHER ABOVE OR BELOW L
      40 IF(I-L) 60,50,50
      50 S(1S)=A(I,J)
      GO TO 70
      60 R(1R)=A(I,J)
      70 CONTINUE
      RETURN
      END
```

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

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

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

```
CTIE 1
CTIE 2
CTIE 3
CTIE 4
CTIE 5
CTIE 6
CTIE 7
CTIE 8
CTIE 9
CTIE 10
CTIE 11
CTIE 12
CTIE 13
CTIE 14
CTIE 15
CTIE 16
CTIE 17
CTIE 18
CTIE 19
CTIE 20
CTIE 21
CTIE 22
CTIE 23
CTIE 24
CTIE 25
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)
C  DIMENSION A(1),R(1)
C  COMPUTE VECTOR LENGTH, IT
C  CALL LOC(N,M,IT,1,MS)
C  COPY MATRIX
DO 1 I=1,IT
1  R(I)=A(I)
RETURN
END
```

```

MCPY 1
MCPY 2
MCPY 3
MCPY 4
MCPY 5
MCPY 6
MCPY 7
MCPY 8
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)
C  DIMENSION A(1),R(1)
C  INITIALIZE
IR=0
L2=L+NR-1
K2=K+MR-1
DO 5 J=K,K2
DO 5 I=L,L2
IR=IR+1
R(IR)=0.0
C  LOCATE ELEMENT FOR ANY MATRIX STORAGE MODE
C  CALL LOC(1,J,1A,NA,MA,MS)
C  TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX
1A(1A) 4,5,4
4 R(IR)=A(1A)
5 CONTINUE
RETURN
END
```

```

XCPY 1
XCPY 2
XCPY 3
XCPY 4
XCPY 5
XCPY 6
XCPY 7
XCPY 8
XCPY 9
XCPY 10
XCPY 11
XCPY 12
XCPY 13
XCPY 14
XCPY 15
XCPY 16
XCPY 17
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)
  DIMENSION A(11,R(1))
  DO 3 J=1,M
    C LOCATE ELEMENT FOR ANY MATRIX STORAGE MODE
    CALL LOC(L,J,L,J,M,MS)
    C TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX
    IF(LJ) 1,2,1
    C MOVE ELEMENT TO R
    1 R(J)=A(LJ)
    GO TO 3
    2 R(J)=0.0
    3 CONTINUE
  RETURN
END
```

```
RCPY 1
RCPY 2
RCPY 3
RCPY 4
RCPY 5
RCPY 6
RCPY 7
RCPY 8
RCPY 9
RCPY 10
RCPY 11
RCPY 12
RCPY 13
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)
  DIMENSION A(11,R(1))
  DO 3 I=1,N
    C LOCATE ELEMENT FOR ANY MATRIX STORAGE MODE
    CALL LOC(I,L,I,L,M,MS)
    C TEST FOR ZERO ELEMENT IN DIAGONAL MATRIX
    IF(IIL) 1,2,1
    C MOVE ELEMENT TO R
    1 R(I)=A(IL)
    GO TO 3
    2 R(I)=0.0
    3 CONTINUE
  RETURN
END
```

```
CCPY 1
CCPY 2
CCPY 3
CCPY 4
CCPY 5
CCPY 6
CCPY 7
CCPY 8
CCPY 9
CCPY 10
CCPY 11
CCPY 12
CCPY 13
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)
  DIMENSION A(1),R(1)
  DO 3 J=1,N
    LOCATE DIAGONAL ELEMENT FOR ANY MATRIX STORAGE MODE
  C   CALL LOC(J,J,1,J,N,MS)
  C   MOVE DIAGONAL ELEMENT TO VECTOR R
  3 R(J)=A(1,J)
  RETURN
END
```

```

DCPY 1
DCPY 2
DCPY 3
DCPY 4
DCPY 5
DCPY 6
DCPY 7
DCPY 8
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)
  DIMENSION A(1)
  C   COMPUTE VECTOR LENGTH, IT
  C   CALL LOC(N,M,IT,N,M,MS)
  C   REPLACE BY SCALAR
  DO 1 I=1,IT
    1 A(I)=C
  RETURN
END
```

```

SCLA 1
SCLA 2
SCLA 3
SCLA 4
SCLA 5
SCLA 6
SCLA 7
SCLA 8
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)
DIMENSION A(1)
DO 3 I=1,N
  LOCATE DIAGONAL ELEMENT FOR ANY MATRIX STORAGE MODE
  CALL LOC(I,I,N,MS)
  REPLACE DIAGONAL ELEMENTS
3 A(I)=C
RETURN
END
```

DCLA 1  
DCLA 2  
DCLA 3  
DCLA 4  
DCLA 5  
DCLA 6  
DCLA 7  
DCLA 8  
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)
DIMENSION A(1),R(1)
DO 20 I=1,N
  DO 20 J=1,N
    IF R IS GENERAL, FORM ELEMENT
    IF(MSR) 5,10,5
  C IF IN LOWER TRIANGLE OF SYMMETRIC OR DIAGONAL R, BYPASS
  C 5 IF(I-J) 10,10,20
  10 CALL LOC(I,J,R,N,MSR)
  C IF IN UPPER AND OFF DIAGONAL OF DIAGONAL R, BYPASS
  C IF(I) 20,20,15
  C OTHERWISE, FORM R(I,J)
  15 R(I)=0
  CALL LOC(I,J,A,N,MSA)
  C IF THERE IS NO A(I,J), LEAVE R(I,J) AT 0.0
  IF(A) 20,20,18
  18 R(I)=A(I)
  20 CONTINUE
RETURN
END
```

MSTR 1  
MSTR 2  
MSTR 3  
MSTR 4  
MSTR 5  
MSTR 6  
MSTR 7  
MSTR 8  
MSTR 9  
MSTR 10  
MSTR 11  
MSTR 12  
MSTR 13  
MSTR 14  
MSTR 15  
MSTR 16  
MSTR 17  
MSTR 18  
MSTR 19  
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)
C DIMENSION A(1),R(1)
C COMPUTE VECTOR LENGTH, IT
C CALL LOC(N,M,IT,N,M,MS)
C BUILD MATRIX R FOR ANY STORAGE MODE
DO 5 I=1,IT
  R(I)=F(A)
5 R(I)=F(A)
RETURN
END
```

MFUN 1  
MFUN 2  
MFUN 3  
MFUN 4  
MFUN 5  
MFUN 6  
MFUN 7  
MFUN 8  
MFUN 9  
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)
BIG=1.0E38
C TEST ELEMENT FOR ZERO
IF(E) 1,2,1
C IF NON-ZERO, CALCULATE RECIPROCAL
1 RECP=1.0/E
RETURN
C IF ZERO, SET EQUAL TO INFINITY
2 RECP=SIGN(BIG,E)
RETURN
END
```

RECP 1  
RECP 2  
RECP 3  
RECP 4  
RECP 5  
RECP 6  
RECP 7  
RECP 8  
RECP 9  
RECP 10  
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)
  IX=I
  JX=J
  IF(MS-1) 10,20,30
10  IRX=N*(JX-1)+IX
  GO TO 36
20  IF(IX-JX) 22,24,24
22  IRX=IX+(JX-JX-JX)/2
  GO TO 36
24  IRX=JX+(IX-IX-IX)/2
  GO TO 36
30  IRX=0
  IF(IX-JX) 36,32,16
32  IR=IX
36  IR=IRX
  RETURN
  END
```

```
LOC 1
LOC 2
LOC 3
LOC 4
LOC 5
LOC 6
LOC 7
LOC 8
LOC 9
LOC 10
LOC 11
LOC 12
LOC 13
LOC 14
LOC 15
LOC 16
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)
  DIMENSION S(1), D(1)
  N1=N-I
  C TEST TYPE OF CONVERSION
  IF (MODE-1) 100, 100, 120
  C CONVERT FROM SINGLE TO DOUBLE DIMENSION
  100 IJ=I+J+1
  NM=N+J+1
  DO 110 K=1, J
    NM=NM-NI
    DO 110 L=1, I
      IJ=LJ+1
      NM=NM-1
  110 D(NM)=S(IJ)
  GO TO 140
  C CONVERT FROM DOUBLE TO SINGLE DIMENSION
  120 IJ=0
  NM=0
  DO 130 K=1, J
    DO 125 L=1, I
      IJ=LJ+1
      NM=NM+1
  125 S(IJ)=D(NM)
  130 NM=NM+1
  140 RETURN
  END

```

```

ARRAY 1
ARRAY 2
ARRAY 3
ARRAY 4
ARRAY 5
ARRAY 6
ARRAY 7
ARRAY 8
ARRAY 9
ARRAY 10
ARRAY 11
ARRAY 12
ARRAY 13
ARRAY 14
ARRAY 15
ARRAY 16
ARRAY 17
ARRAY 18
ARRAY 19
ARRAY 20
ARRAY 21
ARRAY 22
ARRAY 23
ARRAY 24
ARRAY 25
ARRAY 26

```

Sometimes formula (2) is used in the following form:

$$z_j = z_{j+2} - \frac{h}{3} (y_j + 4y_{j+1} + y_{j+2}) \quad (5)$$

Local truncation errors of formulas (1)...(4) are, respectively:

$$R_1 = \frac{1}{24} h^4 y''''(\xi_1) \quad (\xi_1 \in [x_{j-1}, x_{j+1}])$$

$$R_2 = -\frac{1}{90} h^5 y'''''(\xi_2) \quad (\xi_2 \in [x_{j-2}, x_j])$$

$$R_3 = -\frac{3}{80} h^5 y'''''(\xi_3) \quad (\xi_3 \in [x_{j-3}, x_j])$$

$$R_4 = -\frac{1}{144} h^5 y'''''(\xi_4) \quad (\xi_4 \in [x_{j-5}, x_j])$$

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.

## 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:

$$z_i = z(x_i) = \int_a^{x_i} y(x) dx \quad \left\{ \begin{array}{l} (i = 1, 2, \dots, n) \end{array} \right.$$

with  $x_i = a + (i-1)h$

for a table of function values  $y_i$  ( $i = 1, 2, \dots, n$ ), given at equidistant points  $x_i = a + (i-1)h$  ( $i = 1, 2, \dots, n$ ), 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:

$$z_j = z_{j-1} + \frac{h}{3} (1.25 y_{j-1} + 2y_j - 0.25 y_{j+1}) \quad (1)$$

$$z_j = z_{j-2} + \frac{h}{3} (y_{j-2} + 4y_{j-1} + y_j) \quad \text{(Simpson's rule)} \quad (2)$$

$$z_j = z_{j-3} + \frac{3}{8} h (y_{j-3} + 3y_{j-2} + 3y_{j-1} + y_j) \quad (3)$$

(Newton's 3/8 rule)

$$z_j = z_{j-5} + \frac{h}{3} (y_{j-5} + 3.875 y_{j-4} + 2.625 y_{j-3} + 2.625 y_{j-2} + 3.875 y_{j-1} + y_j) \quad (4)$$

[combination of (2) and (3)]

### 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 than 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)
  DIMENSION Y(1),Z(1)
  HT=.3333333*H
  L1=1
  L2=2
  L3=3
  L4=4
  L5=5
  L6=6
  IF(NDIM=5)HT=.1
  NDIM IS GREATER THAN 5. PREPARATIONS OF INTEGRATION LOOP
1 SUM1=Y(L2)+Y(L2)
  SUM1=SUM1+SUM1
  SUM1=HT*(Y(L3)+SUM1+Y(L3))
  AUX1=Y(L4)+Y(L4)
  AUX1=AUX1+AUX1
  AUX1=SUM1+HT*(Y(L3)+AUX1+Y(L3))
  AUX2=HT*(Y(L1)+.875*(Y(L2)+Y(L5))+2.625*(Y(L3)+Y(L4))+Y(L6))
  SUM2=Y(L5)+Y(L5)
  SUM2=SUM2+SUM2
  SUM2=AUX2+HT*(Y(L4)+SUM2+Y(L6))
  Z(L1)=0
  AUX=Y(L3)+Y(L3)
  AUX=AUX+AUX
  Z(L2)=SUM2+HT*(Y(L2)+AUX+Y(L4))
  Z(L3)=SUM1
  Z(L4)=SUM2
  IF(NDIM=6)HT=.05
  NDIM IS GREATER THAN 6. PREPARATIONS OF INTEGRATION LOOP
2 DO 4 I=7,NDIM+2
  SUM1=AUX1
  SUM2=AUX2
  AUX1=Y(I-1)+Y(I-1)
  AUX1=AUX1+AUX1
  AUX1=SUM1+HT*(Y(I-2)+AUX1+Y(I))
  Z(I-2)=SUM1
  IF(I=NDIM)HT=.03
  AUX2=Y(I)+Y(I)
  AUX2=SUM2+AUX2
  AUX2=AUX2+HT*(Y(I-1)+AUX2+Y(I+1))
  Z(I-1)=SUM2
  Z(NDIM)=AUX1
  Z(NDIM+1)=AUX2
  RETURN
6 Z(NDIM)=SUM2
  Z(NDIM+1)=AUX1
  RETURN
END OF INTEGRATION LOOP
7 IF(NDIM=3)HT=.1
  NDIM IS EQUAL TO 4 OR 5
8 SUM2=Y(L2)+Y(L2)
  SUM1=SUM1+SUM1
  SUM1=HT*(Y(L1)+SUM1+Y(L3))
  Z(L1)=0
  AUX1=Y(L3)+Y(L3)
  AUX1=AUX1+AUX1
  Z(L2)=SUM2+HT*(Y(L2)+AUX1+Y(L4))
  IF(NDIM=5)HT=.05
  AUX1=Y(L4)+Y(L4)
  AUX1=SUM1+HT*(Y(L3)+AUX1+Y(L5))
  Z(L3)=SUM1
  Z(L4)=SUM2
  RETURN
NDIM IS EQUAL TO 3
11 SUM1=HT*(.125*(Y(L1)+Y(L2))+.25*(Y(L3))
  SUM2=Y(L2)+Y(L2)
  SUM2=SUM2+SUM2
  Z(L3)=HT*(Y(L1)+SUM2+Y(L3))
  Z(L1)=0
  Z(L2)=SUM1
12 RETURN
END

```

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:

$$y = \int_a^b f(x) dx \quad (1)$$

Successively dividing the interval  $[a, b]$  into  $2^i$  equidistant subintervals ( $i = 0, 1, 2, \dots$ ) and using the following notations:

$$h_i = \frac{b-a}{2^i}; x_{i,k} = a + k \cdot h_i, f_{i,k} = f(x_{i,k})$$

$$(k = 0, 1, 2, \dots, 2^i)$$

the trapezoidal rule gives approximations  $T_{0,i}$  to the integral value  $y$ :

$$T_{0,i} = h_i \left\{ \sum_{k=0}^{2^i} f_{i,k} - \frac{1}{2} (f(a) + f(b)) \right\} \quad (2)$$

Then the following can be written:

$$T_{0,i} = y + \sum_{r=1}^{\infty} C_{0,2r} \cdot h_i^{2r}$$

with unknown coefficients  $C_{0,2r}$  which do not depend on  $i$ . Thus there is a truncation error of the order  $h_i^2$ .

Knowing two successive approximations,  $T_{0,i}$  and  $T_{0,i+1}$ , an extrapolated value can be generated:

$$T_{1,i} = T_{0,i+1} + \frac{T_{0,i+1} - T_{0,i}}{2^2 - 1} \quad (3)$$

This is a better approximation to  $y$  because:

$$T_{1,i} = y + \frac{1}{2^2 - 1} \sum_{r=1}^{\infty} C_{0,2r} (2^2 h_{i+1}^{2r} - h_i^{2r})$$

Noting that  $2^2 h_{i+1}^2 - h_i^2 = 0$  and setting:

$$C_{1,2r} = \frac{1}{2^2 - 1} (2^2 - 2^{2r}) \cdot C_{0,2r}$$

$T_{1,i}$  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  $T_{0,i+2}$  also,  $T_{1,i+1}$  can be generated (formula 3), and:

$$T_{2,i} = T_{1,i+1} + \frac{T_{1,i+1} - T_{1,i}}{2^4 - 1} \quad (4)$$

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:

$$T_{k,j} = T_{k-1,j+1} + \frac{T_{k-1,j+1} - T_{k-1,j}}{2^{2k-1}} \quad (k+j=i, \quad j=i-1, i-2, \dots, 2, 1, 0) \quad (5)$$

and storing:

$T_{0,i}$  into AUX (i+1)

$T_{1,i-1}$  into AUX (i)

$T_{k,0}$  into AUX (1)

| Truncation error |          | $O(h_i^2)$ | $O(h_i^4)$ | $O(h_i^6)$ | $O(h_i^8) \dots$ |
|------------------|----------|------------|------------|------------|------------------|
| step length      | j        | 0          | 1          | 2          | 3 ...            |
| $h_i$            | i        |            |            |            |                  |
| $b-a$            | 0        | $T_{0,0}$  | $T_{1,0}$  | $T_{2,0}$  | $T_{3,0} \dots$  |
| $\frac{b-a}{2}$  | 1        | $T_{0,1}$  | $T_{1,1}$  | $T_{2,1}$  | $\vdots$         |
| $\frac{b-a}{4}$  | 2        | $T_{0,2}$  | $T_{1,2}$  | $\vdots$   |                  |
| $\frac{b-a}{8}$  | 3        | $T_{0,3}$  | $\vdots$   |            |                  |
| $\vdots$         | $\vdots$ | $\vdots$   |            |            |                  |

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, NDM, 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. NDM-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 NDM.

#### 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 NDM is less than 5, or the required accuracy could not be reached within NDM-1 steps. NDM 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 NDM) 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,NDM,FCT,Y,IER,AUX)
  DIMENSION AUX(1)
  PREPARATIONS OF ROMBERG-LOOP
  AUX(1)=S*(FCT(XL)+FCT(XU))
  H=XU-XL
  IF (NDM-1) 18,H+1
  1 IF (H/2) 10,2
  NDM IS GREATER THAN 1 AND H IS NOT EQUAL TO 0.
  2 H=H/2
  E=EPS/ABS(H)
  DELT2=0.
  P=1.
  JJ=1
  DO 7 I=2,NDM
    Y=AUX(I)
    DELT1=DELT2
    HD=HM
    HM=.5*HH
    P=.5*P
    X=XL+HM
    S=AUX(I)
  QATR 1
  QATR 2
  QATR 3
  QATR 4
  QATR 5
  QATR 6
  QATR 7
  QATR 8
  QATR 9
  QATR 10
  QATR 11
  QATR 12
  QATR 13
  QATR 14
  QATR 15
  QATR 16
  QATR 17
  QATR 18
  QATR 19
  QATR 20
  QATR 21

```

```

DO 3 J=1,JJ
SM=SM+FCT(X)
3 X=X+HD
AUX(1)=5*AUX(1)-1)+P*SM
C A NEW APPROXIMATION OF INTEGRAL VALUE IS COMPUTED BY MEANS OF
C TRAPEZOIDAL RULE.
C START OF ROMBERGS EXTRAPOLATION METHOD.
Q=1
J1=1
DO 4 J=1,J1
I1=J
Q=Q+Q
D=Q+Q
4 AUX(11)=AUX(I1+1)+1AUX(I1+1)-AUX(I11)/(Q-1.)
END OF ROMBERG-STEP
DELT2=ABS(Y-AUX(11))
IF(I=9)7,5,5
5 IF(DELT2-E)10,10,6
6 IF(DELT2-DELT1)7,11,11
7 JJ=JJ+JJ
8 IER=2
9 Y=AUX(11)
RETURN
10 IER=0
GO TO 9
11 IER=1
Y=HY
RETURN
END

```

```

QATR 22
QATR 23
QATR 24
QATR 25
QATR 26
QATR 27
QATR 28
QATR 29
QATR 30
QATR 31
QATR 32
QATR 33
QATR 34
QATR 35
QATR 36
QATR 37
QATR 38
QATR 39
QATR 40
QATR 41
QATR 42
QATR 43
QATR 44
QATR 45
QATR 46
QATR 47
QATR 48
QATR 49
QATR 50

```

#### 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.

#### 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:

$$\frac{dy}{dx} = f(x, y) \quad (1)$$

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:

$$y_{n+1} = y_n + 1/6 [k_0 + 2k_1 + 2k_2 + k_3] \quad (2)$$

where we define, for step size  $h$

$$\begin{cases} k_0 = hf(x_n, y_n) \\ k_1 = hf(x_n + h/2, y_n + k_0/2) \\ k_2 = hf(x_n + h/2, y_n + k_1/2) \\ k_3 = hf(x_n + h, y_n + k_2) \end{cases} \quad (3)$$

##### 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)

###### 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)
C IF XF IS LESS THAN OR EQUAL TO XI, RETURN XI, YI AS ANSWER
IF(XF-XI) 11,11,12
11 ANSX=XI
ANSY=YI
RETURN
C TEST INTERVAL VALUE
12 H=HI
IF(H) 10,14,20
14 IER=1
ANSX=XI
ANSY=YI
RETURN
10 H=H/2
SET XN=INITIAL X, YN=INITIAL Y
20 XN=XI
YN=YI
C INTEGRATE ONE TIME STEP
HNEW=H
JUMP=1
GO TO 170
25 XN1=XN
YN1=YN
C COMPARE XN1 (=X(N+1)) TO X FINAL AND BRANCH ACCORDINGLY
IF(XN1-XF)50,30,40
XN1=XF, RETURN (XF, YN1) AS ANSWER
30 ANSX=XF
ANSY=YN1
GO TO 160
C XN1 GREATER THAN XF, SET NEW STEP SIZE AND INTEGRATE ONE STEP
HNEW=H/2
JUMP=2
GO TO 170
45 ANSX=XN
ANSY=YN
GO TO 160
C XN1 LESS THAN X FINAL, CHECK IF (YN, YN1) SPAN Y FINAL
50 IF((YN1-YF)*(YF-YN))>0,70,110

```

```

C      YN1 AND YN DO NOT SPAN YF. SET (XN,YN) AS (XN1,YN1) AND REPEAT RK1 40
60 YN=YN1 41
ENNYN1 42
GO TO 170 43
C      EITHER YN OR YN1 =YF. CHECK WHICH AND SET PROPER (X,Y) AS ANSWER 44
70 IF(YN1-YF)100,100,80 45
80 ANSY=YN 46
ANSX=XN 47
GO TO 160 48
100 ANSY=YN1 49
ANSX=XN1 50
GO TO 160 51
C      YN AND YN1 SPAN YF. TRY TO FIND X VALUE ASSOCIATED WITH YF 52
110 DO 140 I=1,10 53
C      INTERPOLATE TO FIND NEW TIME STEP AND INTEGRATE ONE STEP 54
C      TRY TEN INTERPOLATIONS AT MOST 55
HNEW=(YF-YN)/(YN1-YN)*(XN1-XN) 56
JUMP=3 57
GO TO 170 58
115 XNEW=X 59
YNEW=Y 60
C      COMPARE COMPUTED Y VALUE WITH YF AND BRANCH 61
IF(YNEW-YF)120,150,130 62
C      ADVANCE, YF IS BETWEEN YNEW AND YN1 63
120 YN=YNEW 64
XN=XNEW 65
GO TO 140 66
C      ADVANCE, YF IS BETWEEN YN AND YNEW 67
130 YN1=YNEW 68
XN1=XNEW 69
140 CONTINUE 70
C      RETURN (XNEW,YF) AS ANSWER 71
150 ANSX=XNEW 72
ANSY=YF 73
160 RETURN 74
170 H2=HNEW/2.0 75
T1=HNEW*FUN(XN,YN) 76
T2=HNEW*FUN(XN+H2,YN+T1/2.0) 77
T3=HNEW*FUN(XN+H2,YN+T2/2.0) 78
T4=HNEW*FUN(XN+HNEW,YN+T3) 79
YN=YN+(T1+2.0*T2+2.0*T3+T4)/6.0 80
XX=XN+HNEW 81
GO TO (25,45,115), JUMP 82
END 83

```

## Subroutine RK2

### Purpose:

Integrates a first-order differential equation  $dy/dx = \text{FUN}(x, y)$  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.

## RK2

This subroutine integrates a given function using the Runge-Kutta technique and produces tabulated values of the computed integral.

The ordinary differential equation:

$$\frac{dy}{dx} = f(x, y) \quad (1)$$

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:

$$y_{n+1} = y_n + 1/6 [k_0 + 2k_1 + 2k_2 + k_3] \quad (2)$$

where we define, for step size  $h$

$$\begin{cases} k_0 = hf(x_n, y_n) \\ k_1 = hf(x_n + h/2, y_n + k_0/2) \\ k_2 = hf(x_n + h/2, y_n + k_1/2) \\ k_3 = hf(x_n + h, y_n + k_2) \end{cases} \quad (3)$$

```

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=HFUN(X,Y)  RK2  8
    T2=HFUN(X+H2,Y+T1/2.)  RK2  9
    T3=HFUN(X+H2,Y+T2/2.)  RK2  10
    T4=HFUN(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:

$$\begin{aligned} y_1' &= \frac{dy_1}{dx} = f_1(x, y_1, y_2, \dots, y_n) \\ y_2' &= \frac{dy_2}{dx} = f_2(x, y_1, y_2, \dots, y_n) \\ &\dots\dots\dots \\ y_n' &= \frac{dy_n}{dx} = f_n(x, y_1, y_2, \dots, y_n) \end{aligned}$$

and the initial values:

$$y_1(x_0) = y_{1,0}, y_2(x_0) = y_{2,0}, \dots, y_n(x_0) = y_{n,0}$$

and using the following vector notations:

$$Y(x) = \begin{pmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_n(x) \end{pmatrix}, F(x, Y) = \begin{pmatrix} f_1(x, Y) \\ f_2(x, Y) \\ \vdots \\ f_n(x, Y) \end{pmatrix}, Y_0 = \begin{pmatrix} y_{1,0} \\ y_{2,0} \\ \vdots \\ y_{n,0} \end{pmatrix}$$

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  $Y_4 = Y(x_0 + h)$  is computed by the following formulas:

$$\begin{aligned} K_1 &= hF(x_0, Y_0) \quad ; \quad Y_1 = Y_0 + \frac{1}{2}(K_1 - 2Q_0) \\ Q_1 &= Q_0 + 3\left[\frac{1}{2}(K_1 - 2Q_0)\right] - \frac{1}{2}K_1 \\ K_2 &= hF\left(x_0 + \frac{h}{2}, Y_1\right) \quad ; \quad Y_2 = Y_1 + \left(1 - \sqrt{\frac{1}{2}}\right)(K_2 - Q_1) \\ Q_2 &= Q_1 + 3\left[\left(1 - \sqrt{\frac{1}{2}}\right)(K_2 - Q_1)\right] - \left(1 - \sqrt{\frac{1}{2}}\right)K_2 \\ &\dots\dots\dots (1) \\ K_3 &= hF\left(x_0 + \frac{h}{2}, Y_2\right) \quad ; \quad Y_3 = Y_2 + \left(1 + \sqrt{\frac{1}{2}}\right)(K_3 - Q_2) \\ Q_3 &= Q_2 + 3\left[\left(1 + \sqrt{\frac{1}{2}}\right)(K_3 - Q_2)\right] - \left(1 + \sqrt{\frac{1}{2}}\right)K_3 \\ K_4 &= hF(x_0 + h, Y_3) \quad ; \quad Y_4 = Y_3 + \frac{1}{6}(K_4 - 2Q_3) \\ Q_4 &= Q_3 + 3\left[\frac{1}{6}(K_4 - 2Q_3)\right] - \frac{1}{2}K_4 \end{aligned}$$

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  $\delta$  for accuracy is generated in the following way:

$$\delta = \frac{1}{15} \sum_{i=1}^n a_i \cdot |y_i^{(1)} - y_i^{(2)}| \quad (2)$$

where the coefficients  $a_i$  are error-weights specified in the input of the procedure.

Test value  $\delta$  is an approximate measure for the local truncation error at point  $x_0 + 2h$ . If  $\delta$  is greater than a given tolerance  $\epsilon_2$ , increment  $h$  is halved and the procedure starts again at the point  $x_0$ . If  $\delta$  is less than  $\epsilon_2$ , 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  $\delta$  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  $j = 1, 2, \dots$ ) 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  $\epsilon_2$  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  $\epsilon_2$ , 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  $x_0 + nh$  ( $n = 0, 1, 2, \dots$ ). 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 OUTF. 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 OUTF.
- 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.
- OUTF - 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 OUTF. 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 OUTF has changed PRMT(5) to nonzero.

#### Subroutines and function subprograms required:

The external subroutines FCT(X, Y, DERY) and OUTF(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,OUTF,AUX)
  DIMENSION Y(1),DERY(1),AUX(8,1),A(4),B(4),C(4),PRMT(5)
  DO 1 I=1,NDIM
    1 AUX(I,1)=0.0666667*DERY(I)
    X=PRMT(1)
    XEND=PRMT(2)
    H=PRMT(3)
    PRMT(5)=0.
    CALL FCT(X,Y,DERY)
    ERROR TEST
    IF (H*(XEND-X)/30. > .37, 2
  PREPARATIONS FOR RUNGE-KUTTA METHOD
  2 A(1)=.5
    A(2)=.2928932
    A(3)=1.707107
    A(4)=.1666667
    B(1)=2.
    B(2)=1.
    B(3)=1.
    B(4)=2.
    C(1)=.5
    C(2)=.2928932
    C(3)=1.707107
    C(4)=.5
  PREPARATIONS OF FIRST RUNGE-KUTTA STEP
  DO 3 I=1,NDIM
    AUX(I,1)=Y(I)
    AUX(I,2)=DERY(I)
    AUX(I,3)=0.
    3 AUX(I,4)=0.
    IREC=0
    HNEW=H
    IHLF=1
    ISTEP=0
    IEND=0
  START OF A RUNGE-KUTTA STEP
  4 IF (X+H-XEND) > .17, 6, 5
  5 H=XEND-X
  6 IEND=1
  RECORDING OF INITIAL VALUES OF THIS STEP
  7 CALL OUTF(X,Y,DERY,IREC,NDIM,PRMT)
  IF (PRMT(5) > 0, 8, 40
  8 ITEST=0
  9 ISTEP=ISTEP+1
  START OF INNERMOST RUNGE-KUTTA LOOP
  J=1
  10 A(J)=A(1)
    B(J)=B(1)
    C(J)=C(1)
    DO 11 I=1,NDIM
      RKGS 1
      RKGS M01
      RKGS 3
      RKGS 4
      RKGS 5
      RKGS 6
      RKGS 7
      RKGS 8
      RKGS 9
      RKGS 10
      RKGS 11
      RKGS 12
      RKGS 13
      RKGS 14
      RKGS 15
      RKGS 16
      RKGS 17
      RKGS 18
      RKGS 19
      RKGS 20
      RKGS 21
      RKGS 22
      RKGS 23
      RKGS 24
      RKGS 25
      RKGS 26
      RKGS 27
      RKGS 28
      RKGS 29
      RKGS 30
      RKGS 31
      RKGS 32
      RKGS 33
      RKGS 34
      RKGS 35
      RKGS 36
      RKGS 37
      RKGS 38
      RKGS 39
      RKGS 40
      RKGS 41
      RKGS 42
      RKGS 43
      RKGS 44
      RKGS 45
      RKGS 46
      RKGS 47
      RKGS 48
      RKGS 49
      RKGS 50

```

```

R1=H*DERV(1)
R2=A*(R1-S)*AUX(6:1)
Y(1)=Y(1)+R2
R2=R2+R2+R2
11 AUX(6:1)=AUX(6:1)+R2-CJ*R1
IF(J=1)2,15,15
12 J=J+1
IF(J=3)13,16,13
13 X=X+H
14 CALL PCT(X,Y,DERV)
GOTO 10
C
END OF INNERMOST RUNGE-KUTTA LOOP
C
TEST OF ACCURACY
15 IF(1TEST)16,16,20
IN CASE 1TEST=0 THERE IS NO POSSIBILITY FOR TESTING OF ACCURACY
16 DO 17 I=1,NDIM
17 AUX(4:1)=Y(I)
1TEST=1
1STEP=1STEP+1STEP=2
18 1MLF=1MLF+1
X=X+H
H=H*50H
DO 19 I=1,NDIM
Y(I)=AUX(1:1)
DERV(I)=AUX(2:1)
19 AUX(6:1)=AUX(3:1)
GOTO 9
C
IN CASE 1TEST=1 TESTING OF ACCURACY IS POSSIBLE
20 1MOD=1STEP/2
IF(1STEP-1MOD=1MOD)21,23,21
21 CALL PCT(X,Y,DERV)
DO 22 I=1,NDIM
AUX(5:1)=Y(I)
22 AUX(7:1)=DERV(I)
GOTO 9
C
COMPUTATION OF TEST VALUE DELT
23 DELT=0
DO 24 I=1,NDIM
24 DELT=DELT+AUX(8:1)+ABS(AUX(4:1)-Y(I))
IF(DELT-PRMT(4))28,28,25
C
ERROR IS TOO GREAT
25 IF(1MLF=10)26,36,36
26 DO 27 I=1,NDIM
27 AUX(4:1)=AUX(3:1)
1STEP=1STEP+1STEP=4
X=X+H
1END=0
GOTO 18
C
RESULT VALUES ARE GOOD
28 CALL PCT(X,Y,DERV)
DO 29 I=1,NDIM
AUX(1:1)=Y(I)
AUX(2:1)=DERV(I)
AUX(3:1)=AUX(6:1)
Y(I)=AUX(5:1)
29 DERV(I)=AUX(7:1)
CALL OUTP(X=H,Y=DERV,1MLF,NDIM,PRMT)
IF(PRMT(5))40,30,40
30 DO 31 I=1,NDIM
Y(I)=AUX(1:1)
31 DERV(I)=AUX(2:1)
JREC=1MLF
IF(1END)32,32,39
INCREMENT GETS DOUBLED
32 1MLF=1MLF+1
1STEP=1STEP/2
H=H*H
IF(1MLF)4,33,33
33 1MOD=1STEP/2
IF(1STEP-1MOD=1MOD)4,34,4
34 IF(DELT-OS+PRMT(4))35,35,4
35 1MLF=1MLF+1
1STEP=1STEP/2
H=H*H
GOTO 4
C
RETURNS TO CALLING PROGRAM
36 1MLF=1
CALL PCT(X,Y,DERV)
GOTO 39
37 1MLF=12
GOTO 39
38 1MLF=13
CALL OUTP(X,Y,DERV,1MLF,NDIM,PRMT)
40 RETURN
END

```

RKGS 51  
 RKGS 52  
 RKGR 53  
 RKGS 54  
 RKGS 55  
 RKGS 56  
 RKGS 57  
 RKGS 58  
 RKGS 59  
 RKGS 60  
 RKGS 61  
 RKGS 62  
 RKGS 63  
 RKGS 64  
 RKGS 65  
 RKGS 66  
 RKGS 67  
 RKGS 68  
 RKGS 69  
 RKGS 70  
 RKGS 71  
 RKGS 72  
 RKGS 73  
 RKGS 74  
 RKGS 75  
 RKGS 76  
 RKGS 77  
 RKGS 78  
 RKGS 79  
 RKGS 80  
 RKGS 81  
 RKGS 82  
 RKGS 83  
 RKGS 84  
 RKGS 85  
 RKGS 86  
 RKGS 87  
 RKGS 88  
 RKGS 89  
 RKGS 90  
 RKGS 91  
 RKGS 92  
 RKGS 93  
 RKGS 94  
 RKGS 95  
 RKGS 96  
 RKGS 97  
 RKGS 98  
 RKGS 99  
 RKGS 100  
 RKGS 101  
 RKGS 102  
 RKGS 103  
 RKGS 104  
 RKGS 105  
 RKGS 106  
 RKGS 107  
 RKGS 108  
 RKGS 109  
 RKGS 110  
 RKGS 111  
 RKGS 112  
 RKGS 113  
 RKGS 114  
 RKGS 115  
 RKGS 116  
 RKGS 117  
 RKGS 118  
 RKGS 119  
 RKGS 120  
 RKGS 121  
 RKGS 122  
 RKGS 123  
 RKGS 124  
 RKGS 125  
 RKGS 126  
 RKGS 127  
 RKGS 128  
 RKGS 129  
 RKGS 130  
 RKGS 131  
 RKGS 132  
 RKGS 133  
 RKGS 134  
 RKGS 135

$$S_1 = \sin\left(\frac{2\pi}{2N+1}\right) \quad (2)$$

$$U_2 = 0$$

$$U_1 = 0$$

$$C = 1$$

$$S = 0$$

$$J = 1$$

The following recursive sequence is used to compute  $U_0$ ,  $U_1$ , and  $U_2$ :

$$U_0 = f\left(\frac{2m\pi}{2N+1}\right) + 2C U_1 - U_2 \quad (3)$$

$$U_2 = U_1$$

$$U_1 = U_0$$

for values of  $m = 2N, 2N-1, \dots, 1$

The coefficients are then:

$$A_J = \frac{2}{2N+1} \left( f(0) + C U_1 - U_2 \right) \quad (4)$$

$$B_J = \frac{2}{2N+1} S U_1 \quad (5)$$

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  $F(X) = A(0) + \text{SUM}(A(K)\cos KX + B(K)\sin KX)$  where  $K=1, 2, \dots, M$  to approximate the computed values of a given function subprogram.

## 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  $2\pi$
  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:

$$C_1 = \cos\left(\frac{2\pi}{2N+1}\right) \quad (1)$$

# 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.,  $A_0, \dots, A_M$ .
- B - Resultant vector of Fourier sine coefficients of length  $M+1$ ; i.e.,  $B_0, \dots, B_M$ .
- 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)
  DIMENSION A(1),B(1)
  CHECK FOR PARAMETER ERRORS
  IER=0
  20 IF(N) 30,40,40
  30 IER=2
  RETURN
  40 IF(M-N) 60,60,50
  50 IER=1
  RETURN
  C COMPUTE AND PRESET CONSTANTS
  60 AN=M
  COEF=2.0/(2.0*AN+1.0)
  CONST=.141593*COEF
  S1=SIN(CONST)
  C1=COS(CONST)
  C=1.0
  S=0.0
  J=1
  FUNZ=FUN(0.0)
  70 U2=0.0
  U1=0.0
  A1=FUNZ
  FOR FOURIER COEFFICIENTS RECURSIVELY
  75 X=A1*CONST
  U0=FUN(X)+2.0*C*J*U2
  U1=U0
  A1=A1-1.0
  IF(A1) 80,80,75
  80 A(J)=COEF*(FUNZ+C*U1-U2)

```

```

B(J)=COEF*S*U1
  IF(J-(M+1)) 90,120,100
  90 Q=C1*C-S1*S
  S=C1*S+S1*C
  C=Q
  J=J+1
  GO TO 70
  100 A(1)=A(1)*0.5
  RETURN
  END

```

```

FORIF 32
FORIF 33
FORIF 34
FORIF 35
FORIF 36
FORIF 37
FORIF 38
FORIF 39
FORIF 40
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  $2\pi$  in steps of  $2\pi/(2N+1)$
  2. N such that there are  $2N+1$  tabulated data points:  $2K\pi/2N+1$ ,  $K = 0, 1, 2, \dots, 2N$
  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  $U_2$ :

$$U_0 = f\left(\frac{2m\pi}{2N+1}\right) + 2C U_1 - U_2 \quad (3)$$

$$U_2 = U_1$$

$$U_1 = U_0$$

for values of  $m = 2N, 2N-1, \dots, 1$

The coefficients are then:

$$A_J = \frac{2}{2N+1} (f(0) + C U_1 - U_2) \quad (4)$$

$$B_J = \frac{2}{2N+1} S U_1 \quad (5)$$

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  $F(X) = A(0) + \text{SUM}(A(K)\cos KX + B(K)\sin KX)$  where  $K=1, 2, \dots, M$  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.,  $A_0, \dots, A_M$ .
- B - Resultant vector of Fourier sine coefficients of length  $M+1$ ; i.e.,  $B_0, \dots, B_M$ .
- 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)
  DIMENSION A(1),B(1),FNT(1)
  C CHECK FOR PARAMETER ERRORS
  IER=0
  20 IF(M) 30,40,40
  30 IER=2
  RETURN
  40 IF(M=N) 60,60,50
  50 IER=1
  RETURN
  C COMPUTE AND PRESET CONSTANTS
  60 AN=N
  COEF=2.0/12.0*AN*1.01
  CONST=3.141593*COEF
  S1=SIN(CONST)
  C1=COS(CONST)
  C=L.0
  S=0.0
  J=1
  FNTZ=FNT(1)
  70 U2=0.0
  U1=0.0
  I=2*N+1
  C FORM FOURIER COEFFICIENTS RECURSIVELY
  75 UD=FNT(I)+2.0*C*J1-U2
  U2=U1
  U1=UD
  I=I-1
  IF(I-1) 80,80,75
  80 A(I)=COEF*(FNTZ+C*U1-U2)
  B(I)=COEF*S*U1
  IF(I-1) 90,130,100
  90 O=C1*U1-S1*U2
  S=C1*S+S1*C
  C=O
  J=J+1
  GO TO 70
  100 A(1)=A(I)+0.5
  RETURN
  ENR
FORIT 1
FORIT 2
FORIT 3
FORIT 4
FORIT 5
FORIT 6
FORIT 7
FORIT 8
FORIT 9
FORIT 10
FORIT 11
FORIT 12
FORIT 13
FORIT 14
FORIT 15
FORIT 16
FORIT 17
FORIT 18
FORIT 19
FORIT 20
FORIT 21
FORIT 22
FORIT 23
FORIT 24
FORIT 25
FORIT 26
FORIT 27
FORIT 28
FORIT 29
FORIT 30
FORIT 31
FORIT 32
FORIT 33
FORIT 34
FORIT 35
FORIT 36
FORIT 37
FORIT 38
FORIT 39
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:

$$\Gamma(x) = \int_0^{\infty} t^{x-1} \cdot e^{-t} dt \quad (1)$$

This function satisfies the recurrence relation:

$$\Gamma(x) = (x-1) \cdot \Gamma(x-1) \quad (2)$$

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  $x > 1$

$$\Gamma(x) = (x-1)(x-2) \dots (x-r) \Gamma(y) \quad (3)$$

$\Gamma(y)$  is computed from the following formula:

$$\begin{aligned} \Gamma(y) \approx & 1 - 0.57710166(y-1) + 0.98585399(y-1)^2 \\ & - 0.87642182(y-1)^3 + 0.83282120(y-1)^4 \end{aligned}$$

$$- 0.56847290(y-1)^5 + 0.25482049(y-1)^6$$

$$- 0.05149930(y-1)^7 \quad (4)$$

For  $x < 1$ , the recurrence (2) is taken in the direction of decreasing  $n$ , giving

$$\Gamma(x) = \frac{\Gamma(y)}{x(x+1)(x+2)\dots(x+r-1)} \quad (5)$$

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)
  IF (XX=34.51616) 4
  IER=2
  GX=1.E38
  RETURN
  6 X=XX
  ERR=1.0E-6
  IER=0
  GX=1.0
  IF (X-2.015015015) 10
  10 IF (X-2.011011011) 15
  15 X=X-1.0
  GX=GX*X
  GO TO 10
  50 IF (X-1.0160120110)
  C   SEE IF X IS NEAR NEGATIVE INTEGER OR ZERO
  60 IF (X-ERR) 62, 62, 80
  62 K=X
  Y=FLOAT(K)-X
  IF (ABS(Y)-ERR) 130, 130, 64
  64 IF (1.0-Y-ERR) 130, 130, 70
  C   X NOT NEAR A NEGATIVE INTEGER OR ZERO
  70 IF (X-1.0180180110)
  80 GX=GX/Y
  X=X-1.0
  GO TO 70
  110 Y=X-1.0
  GY=1.0+Y*(1.0+5771017+Y*(1.0+9858540+Y*(1.0+8766218+Y*(1.0+8328212+
  1Y*(-0.5684729+Y*(1.0+2548205+Y*(-0.05149930))))))
  GX=GX*GY
  120 RETURN

```

```

GAMMA 1
GAMMAH01
GAMMAH02
GAMMAH03
GAMMAH04
GAMMAH05
GAMMA 3
GAMMA 4
GAMMA 5
GAMMA 6
GAMMA 7
GAMMA 8
GAMMA 9
GAMMA 10
GAMMA 11
GAMMA 12
GAMMA 13
GAMMA 14
GAMMA 15
GAMMA 16
GAMMA 17
GAMMA 18
GAMMA 19
GAMMA 20
GAMMA 21
GAMMA 22
GAMMA 23
GAMMA 24
GAMMA 25
GAMMA 26
GAMMA 27

```

```

130 IER=1
RETURN
END

```

```

GAMMA 28
GAMMA 29
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  $|x| = 1$ :

$$e_{n-1} = 0, e_n = e, |e_{n+1}| = 2e, \dots, |e_{n+r}| = (r+1)e$$

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)
  DIMENSION Y(1)
  TEST OF ORDER
  L1=1
  L2=2
  Y(L1)=1.0
  IF(N1)1,2
1  RETURN
  Y(L2)=X
  IF(N=1)1,3
  DO 4 1=2,N
    G=X*Y(1)
    Y(1+1)=G-Y(1-1)  -(G-Y(1-1))/FLOAT(1)+G
  RETURN
END

```

```

LEP  1
LEP  2
LEP  3
LEP  M01
LEP  M02
LEP  M03
LEP  5
LEP  6
LEP  M04
LEP  8
LEP  9
LEP  10
LEP  11
LEP  12
LEP  13

```

# BESJ

This subroutine computes the J Bessel function for a given argument and integer order by using the recurrence relationship:

$$F_{n+1}(x) + F_{n-1}(x) = \left( \frac{2n}{x} \right) F_n(x) \quad (1)$$

The desired Bessel function is:

$$J_n(x) = \frac{F_n(x)}{\alpha} \quad (2)$$

where

$$\alpha = F_0(x) + 2 \sum_{m=1}^{M-2} F_{2m}(x) \quad (3)$$

M is initialized at  $M_0$ .

$M_0$  is the greater of  $M_A$  and  $M_B$  where:

$$M_A = [x+6] \text{ if } x < 5 \text{ and } M_A = [1.4x+60/x] \text{ if}$$

$$x \geq 5.$$

$$M_B = [n+x/4+2]$$

$F_{M-2}, F_{M-3}, \dots, F_2, F_1, F_0$  is evaluated using equation (1) with  $F_M = 0$  and  $F_{M-1} = 10^{-30}$ .

$\alpha$  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:

$$M_{MAX} = \begin{cases} \left[ 20 + 10x - \frac{x^2}{3} \right] & \text{for } x \leq 15 \\ [90 + x/2] & \text{for } x > 15 \end{cases} \quad (4)$$

## 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
  - $20 + 10 \cdot X - X^{2/3}$  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)
  BJ=0
  IF(N)10,20,20
  IER=1
  RETURN
20 IF(X)30,30,31
  IER=2
  RETURN
31 IF(X-15.132,32,36
32 NTEST=20.10.0X-X** 2/3
  GO TO 36
34 NTEST=90.0X/2.
36 IF(N-NTEST)40,38,38
  IER=4
  RETURN
40 IER=0
  N1=N+1
  BPREV=0
  C      COMPUTE STARTING VALUE OF *
  IF(X-5.150,60,60
  50 MA=X*6.
  GO TO 70
  60 MA=1.4*X*60./X
  70 MB=N*IFIX(X)/4+2
  NZERO=MA
  IF(MA-MR)80,90,90
  80 NZERO=MB
```

```

  BESJ 1
  BESJ 2
  BESJ 3
  BESJ 4
  BESJ 5
  BESJ 6
  BESJ 7
  BESJ 8
  BESJ 9
  BESJ 10
  BESJ 11
  BESJ 12
  BESJ 13
  BESJ 14
  BESJ 15
  BESJ 16
  BESJ 17
  BESJ 18
  BESJ 19
  BESJ 20
  BESJ 21
  BESJ 22
  BESJ 23
  BESJ 24
  BESJ 25
  BESJ 26
  BESJ 27
```

```

C      SET UPPER LIMIT OF M
  90 MMAX=NTEST
  100 DO 190 M=NZERO,MMAX,3
  C      SET F(M),F(M-1)
  FM1=1.0E-28
  FM=0
  ALPHA=0
  IF(M-1M/2)*2)110,110,120
  110 JT=1
  GO TO 130
  120 JT=1
  130 M2=M-2
  DO 140 K=1,M2
  MK=M-K
  BMK=2.0*FLD(AT(MK)*FM1/X-FM
  FM=FM1
  FM1=BMK
  IF(MK-N-1)150,140,150
  140 BJ=BMK
  150 JT=-JT
  S=1+JT
  160 ALPHA=ALPHA+BMK*S
  BMK=2.0*FM1/X-FM
  IF(N)180,170,180
  170 BJ=BMK
  180 ALPHA=ALPHA+BMK
  BJ=BJ/ALPHA
  IF(ABS(BJ-BPREV)-ABS(0*BJ))200,200,190
  190 BPREV=BJ
  IER=3
  200 RETURN
  END
```

```

  RESJ 28
  RESJ 29
  RESJ 30
  RESJ 31
  RESJ 32
  RESJ 33
  RESJ 34
  RESJ 35
  RESJ 36
  RESJ 37
  RESJ 38
  RESJ 39
  RESJ 40
  RESJ 41
  RESJ 42
  RESJ 43
  RESJ 44
  RESJ 45
  RESJ 46
  RESJ 47
  RESJ 48
  RESJ 49
  RESJ 50
  RESJ 51
  RESJ 52
  RESJ 53
  RESJ 54
  RESJ 55
  RESJ 56
  RESJ 57
  RESJ 58
  RESJ 59
```

### BESY

This subroutine computes the Y Bessel function for a given argument x and order n. The recurrence relation:

$$Y_{n+1}(x) = \left(\frac{2n}{x}\right) \cdot Y_n(x) - Y_{n-1}(x) \quad (1)$$

is used for this evaluation.

For  $x > 4$

$$Y_0(x) = \sqrt{\frac{2}{\pi x}} \left( P_0(x) \sin\left(x - \frac{\pi}{4}\right) + Q_0(x) \cos\left(x - \frac{\pi}{4}\right) \right) \quad (2)$$

$$Y_1(x) = \sqrt{\frac{2}{\pi x}} \left( -P_1(x) \cos\left(x - \frac{\pi}{4}\right) + Q_1(x) \sin\left(x - \frac{\pi}{4}\right) \right) \quad (3)$$

$P_0(x)$ ,  $Q_0(x)$ ,  $P_1(x)$ , and  $Q_1(x)$  are:

$$\begin{aligned} \frac{1}{\sqrt{2\pi}} P_0\left(\frac{4}{t}\right) = & 0.3989422793 - 0.0017530620t^2 \\ & + 0.0001734300t^4 - 0.0000487613t^6 \\ & + 0.0000173565t^8 - 0.0000037043t^{10} \end{aligned} \quad (4)$$

$$\begin{aligned} \frac{1}{t\sqrt{2\pi}} Q_0\left(\frac{4}{t}\right) = & -0.124669441 + 0.0004564324t^2 \\ & - 0.0000869791t^4 + 0.0000342468t^6 \\ & - 0.0000142078t^8 + 0.0000032312t^{10} \end{aligned} \quad (5)$$

$$\begin{aligned} \frac{1}{\sqrt{2\pi}} P_1\left(\frac{4}{t}\right) = & 0.3989422819 + 0.0029218256t^2 \\ & - 0.0002232030t^4 + 0.0000580759t^6 \\ & - 0.0000200920t^8 + 0.0000042414t^{10} \end{aligned} \quad (6)$$

$$\begin{aligned} \frac{1}{t\sqrt{2\pi}} Q_1\left(\frac{4}{t}\right) = & 0.0374008364 - 0.0006390400t^2 \\ & + 0.0001064741t^4 - 0.0000398708t^6 \\ & + 0.0000162200t^8 - 0.0000036594t^{10} \end{aligned} \quad (7)$$

where  $t = \frac{4}{x}$

For  $x \leq 4$

$$Y_0(x) = \frac{2}{\pi} \sum_{m=0}^{15} (-1)^m \left(\frac{x}{2}\right)^{2m} \frac{1}{(m!)^2} \left[ \log \frac{x}{2} + \gamma - H_m \right] \quad (8)$$

where

$$H_m = \sum_{r=1}^m \frac{1}{r} \text{ if } m \geq 1 = 0 \text{ if } m = 0 \quad (9)$$

and  $\gamma$  = Euler's constant = 0.5772156649

$$\begin{aligned} Y_1(x) = & -\frac{2}{\pi x} + \frac{2}{\pi} \sum_{m=1}^{16} (-1)^{m+1} \left(\frac{x}{2}\right)^{2m-1} \\ & \frac{1}{m! (m-1)!} \cdot \left[ \log \frac{x}{2} + \gamma - H_m + \frac{1}{2m} \right] \end{aligned} \quad (10)$$

### 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)
C      CHECK FOR ERRORS IN N AND X
  IF(N180,10,10)
  IER=0
  IF(X1190,190,20)
C      BRANCH IF X LESS THAN OR EQUAL 4
  20 IF(X=0,0140,40,30)
C      COMPUTE Y0 AND Y1 FOR X GREATER THAN 4
  30 T1=X/2
  T2=T1*2
  P0=(((1-0000037063)*T2+.0000173565)*T2-.0000487613)*T2
  1  +.00017343)*T2-.001753062)*T2+.3989423
  Q0=(((1-0000032312)*T2-.0000142078)*T2+.0000342468)*T2
  1  -.0000868791)*T2+.0004566324)*T2-.012466394
  P1=(((1-0000064241)*T2-.0000200920)*T2+.0000580759)*T2
  1  -.000273203)*T2+.0029218261)*T2+.3989423
  Q1=(((1-0000036594)*T2+.00001622)*T2-.0000398708)*T2
  1  +.0001064741)*T2-.0006390490)*T2+.03740084
  A=2.0/SQRT(X)
  B=A*T1
  C=X-.7883982
  Y0=A*Q0*SIN(C)+R*Q0*COS(C)
  Y1=-A*P1*COS(C)+B*Q1*SIN(C)
  GO TO 90
C      COMPUTE Y0 AND Y1 FOR X LESS THAN OR EQUAL TO 4
  40 XX=X/2
  X2=XX*XX
  T=ALOG(XX)+.5772157
  SUM=0
  TERM=T
  Y0=T
  DO TO L=1,15
  IF(L-1)50,60,50
  50 SUM=SUM+1./FLOAT(L-1)
  60 FL=L
  TS=T-SUM
  TERM=(TERM*(-X2)/FL**2)*((1.-1./FL*TS))
  70 Y0=Y0+TERM
  TFRM = XX*(T+.5)
  SUM=0
  Y1=TERM
  DO RO L=2,16
  SUM=SUM+1./FLOAT(L-1)
  FL=L
  FL1=FL-1
  TS=T-SUM
  TERM=(TERM*(-X2)/(FL1*FL))*((TS+.5/FL)/(TS+.5/FL1))
  80 Y1=Y1+TERM
  P12=.6366198
  Y0=P12*Y0
  Y1=-P12/X+P12*Y1

```

```

C      CHECK IF ONLY Y0 OR Y1 IS DESIRED
  90 IF(N=1)100,100,130
C      RETURN EITHER Y0 OR Y1 AS REQUIRED
  100 IF(N110,120,110
  110 BY=Y1
  GO TO 170
  120 BY=Y0
  GO TO 170
C      PERFORM RECURRENCE OPERATIONS TO FIND YN(X)
  130 YA=Y0
  YB=Y1
  K=1
  T=FLOAT(2*K)/X
  YC=TYB-YA
  IF(ABS(YC)-1.0F36)145,145,141
  141 IER=3
  RETURN
  145 K=K+1
  IF(K-N)150,160,140
  150 YA=YB
  YB=YC
  GO TO 140
  160 BY=YC
  170 RETURN
  180 IER=1
  RETURN
  190 IER=2
  RETURN
  END

```

```

BESY 79
BESY 80
BESY 81
BESY 82
BESY 83
BESY 84
BESY 85
BESY 86
BESY 87
BESY 88
BESY 89
BESY 90
BESY 91
BESY 92
BESY 93
BESY 94
BESY 95
BESY 96
BESY 97
BESY 98
BESY 99
BESY 100
BESY 101
BESY 102
BESY 103
BESY 104

```

## BESI

This subroutine computes the I Bessel function for a given argument x and order n.

For  $x \leq 12$  or  $\leq n$

$$I_n(x) = \left(\frac{x}{2}\right)^n \frac{1}{n!} \sum_{s=0}^{30} \left(\frac{x}{2}\right)^{2s} \frac{n!}{s!(n+s)!} \quad (1)$$

For  $x > 12$  and  $> n$

$$I_n(x) = \frac{e^x}{\sqrt{2\pi x}} \sum_{m=0}^{30} (8x)^{-m} \cdot \frac{1}{m!} \prod_{K=1}^m \left( (2K-1)^2 - 4n^2 \right) \quad (2)$$

## 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  $i^{\text{th}}$  Bessel function using series or asymptotic approximations depending on the range of the arguments.

```

SUBROUTINE BESI(X,N, BI,IER)
C
C   CHECK FOR ERRORS IN N AND X AND EXIT IF ANY ARE PRESENT
IER=0
BI=1.0
IF(N)150,15,10
10 IF(X)160,20,20
15 IF(X)180,17,20
17 RETURN
C
C   DEFINE TOLERANCE
20 TOL=1.E-6
C
C   IF ARGUMENT GT 12 AND GT N, USE ASYMPTOTIC FORM
IF(X-12,140,40,30)
30 IF(X-140,40,110)
C
C   COMPUTE FIRST TERM OF SERIES AND SET INITIAL VALUE OF THE SUM
40 XX=X/2.
50 TERM=1.0
IF(N) 70,70,55
55 DO 60 I=1,N
FI=1
IF(ABS(TERM)-1.E-36)56,60,60
56 IER=1
BI=0.0
RETURN
60 TERM=TERM*XX/FI
70 BI=TERM
XX=XX*XX
C
C   COMPUTE TERMS, STOPPING WHEN ABS(TERM) LE ABS(SUM OF TERMS)
C   TIMES TOLERANCE
DO 90 K=1,1000
IF(ABS(TERM)-ABS(BI+TOL))100,100,80
80 FK=X*(N+K)
TERM=TERM*(XX/FK)
90 BI=BI+TERM
RETURN BI AS ANSWER
100 RETURN
C
C   X GT 12 AND X GT N, SO USE ASYMPTOTIC APPROXIMATION
110 FM=X*NNNN
IF(X- 60,0)115,111,111
111 IER=4
RETURN
115 XX=1./(R.*X)
TERM=1.
BI=1.
DO 130 K=1,30
IF(ABS(TERM)-ABS(TOL*BI))140,140,120
120 FK=(2*K-1)**2
TERM=TERM*XX*(FK-FM)/FLOAT(K)
130 BI=BI+TERM
C
C   SIGNIFICANCE LOST AFTER 30 TERMS, TRY SERIES
GO TO 40
140 PI=3.141592653
BI=BI*EXP(X)/SQRT(2.*PI*X)
GO TO 100
150 IER=1
GO TO 100
160 IER=2
GO TO 100
END
BESI 1
BESI 2
BESI 3
BESI 4
BESI 5
BESI 6
BESI 7
BESI 8
BESI 9
BESI 10
BESI 11
BESI 12
BESI 13
BESI 14
BESI 15
BESI M01
BESI M02
BESI M03
BESI M04
BESI M05
BESI M06
BESI M07
BESI M08
BESI M09
BESI M10
BESI 23
BESI M11
BESI M12
BESI M13
BESI 27
BESI 28
BESI 29
BESI 30
BESI 31
BESI 32
BESI 33
BESI 34
BESI M14
BESI M15
BESI M16
BESI M17
BESI 36
BESI 37
BESI 38
BESI 39
BESI 40
BESI 41
BESI 42
BESI M18
BESI M19
BESI 43
BESI 44
BESI 45
BESI 46
BESI 47
BESI 48
BESI 49
BESI 50
```

## BESK

This subroutine computes the K Bessel function for a given argument x and order n.

The recurrence relation:

$$K_{n+1}(x) = \frac{2n}{x} K_n(x) + K_{n-1}(x) \quad (1)$$

is used for this evaluation.

The initial values  $K_0$  and  $K_1$  are found as follows:

For  $x > 1$

$$K_0(x) = e^{-x} \sqrt{\frac{\pi}{2x}} G_0(x) \quad (2)$$

$$K_1(x) = e^{-x} \sqrt{\frac{\pi}{2x}} G_1(x) \quad (3)$$

where  $x = 1/t$  for  $t < 1$

$$\begin{aligned} G_0\left(\frac{1}{t}\right) \sqrt{\frac{\pi}{2}} = & 1.2533141373 - 0.1566641816t \\ & + 0.0881112782t^2 - 0.0913909546t^3 \\ & + 0.1344596228t^4 - 0.2299850328t^5 \\ & + 0.3792409730t^6 - 0.5247277331t^7 \\ & + 0.5575368367t^8 - 0.4262632912t^9 \\ & + 0.2184518096t^{10} - 0.0668097672t^{11} \\ & + 0.0091893830t^{12} \end{aligned} \quad (4)$$

$$\begin{aligned} G_1\left(\frac{1}{t}\right) \sqrt{\frac{\pi}{2}} = & 1.2533141373 + 0.4699927013t \\ & - 0.1468582957t^2 + 0.1280426636t^3 \\ & - 0.1736431637t^4 + 0.2847618149t^5 \\ & - 0.4594342117t^6 + 0.6283380681t^7 \\ & - 0.6632295430t^8 + 0.5050238576t^9 \\ & - 0.2581303765t^{10} + 0.0788000118t^{11} \\ & - 0.0108241775t^{12} \end{aligned} \quad (5)$$

For  $x \leq 1$

$$\gamma = \text{Euler's constant} = 0.5772156649 \quad (6)$$

$$K_0(x) = -\left(\gamma + \log \frac{x}{2}\right) + \sum_{s=1}^6 \left(\frac{x}{2}\right)^{2s} \frac{1}{(s!)^2} \quad (7)$$

$$\left[ H_s - \left(\gamma + \log \frac{x}{2}\right) \right]$$

where

$$H_s = \sum_{r=1}^s \frac{1}{r} \quad (8)$$

$$K_1(x) = \frac{1}{x} + \sum_{s=1}^8 \left(\frac{x}{2}\right)^{2s-1} \frac{1}{(s!)^2} \quad (9)$$
$$\left[ \frac{1}{2} + s \cdot \left(\gamma + \log \frac{x}{2} - H_s\right) \right]$$

## 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  $N^{\text{th}}$  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.

## CEL1

This subroutine computes the complete elliptic integral of the first kind. This is defined as:

$$K(k) = \int_0^{\pi/2} \frac{dt}{\sqrt{1-k^2 \sin^2 t}} \quad 0 \leq k < 1$$

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:

$$k_c^2 + k^2 = 1, \quad 0 < k_c^2 \leq 1$$

The subroutine CEL1 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:

$$a_0 = 1, \quad g_0 = k_c$$

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  $N^{\text{th}}$  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.

$$\text{The value of } K(k) = \frac{\pi}{2 a_N}.$$

## Subroutine CEL1

Purpose:

Calculate complete elliptic integral of first kind.

Usage:

CALL CEL1 (RES, AK, IER)

Mathematics — Special Operations and Functions 105

```

SUBROUTINE BESK(X,N,BK,IER)
  DIMENSION T(12)
  AK=0
  IF(N)10,11,11
10  IER=1
  RETURN
11  IF(X)12,12,20
12  IER=2
  RETURN
20  IF(X= 60.0122,22,21
21  IER=3
  RETURN
22  IER=0
  IF(X=1.136+36,25
25  A=EXP(-X)
  B=1./X
  C=SQRT(B)
  T(1)=B
  DO 26 L=2,12
26  T(L)=T(L-1)*B
  IF(N=1127,29,27
  COMPUTE K0 USING POLYNOMIAL APPROXIMATION
27  G0=A*(1.25331414+15666418*T(1)+088111278*T(2)+091390954*T(3)
  2+13445662*T(4)+22998103*T(5)+37924097*T(6)+52472773*T(7)
  3+55753684*T(8)+42626329*T(9)+21845181*T(10)+066809767*T(11)
  4+009189383*T(12))*C
  IF(N)20,28,29
28  BK=GO
  RETURN
  COMPUTE K1 USING POLYNOMIAL APPROXIMATION
29  G1=A*(1.2533141+16666418*T(1)+12804266*T(3)
  2+17364316*T(4)+28476181*T(5)+45943421*T(6)+62833807*T(7)
  3+66322954*T(8)+50502386*T(9)+25813038*T(10)+078800012*T(11)
  4+010282177*T(12))*C
  IF(N=1120,30,31
  IF(N=1120,30,31
30  BK=G1
  RETURN
  FROM K0,K1 COMPUTE KN USING RECURRENCE RELATION
31  DO 35 J=2,N
  GJ=2.*(FLOAT(J)-1.)*G1/X+GO
  IF(GJ=1.0F36)33,33,32
32  IER=4
  GO TO 34
33  G0=G1
35  G1=GJ
34  BK=GJ
  RETURN
36  B=X/2.
  A=57721566+ALOG(B)
  C=B*B
  IF(N=117,43,37
  COMPUTE K0 USING SERIES EXPANSION
37  G0=A-A
  X2=X*2.
  FACT=1.
  MJ=0
  DO 40 J=1,6
  RJ=1./FLOAT(J)
  X2=X2*JC
  FACT=FACT*RJ*RJ
  MJ=MJ+RJ
  G0=G0+X2*J*FACT*(HJ-A)
  IF(N)43,42,43
42  BK=GO
  RETURN
  COMPUTE K1 USING SERIES EXPANSION
43  X2=X*B
  FACT=1.
  MJ=1.
  G1=1./X+X2*J*(.5+A-MJ)
  DO 50 J=2,8
  X2=X2*JC
  RJ=1./FLOAT(J)
  FACT=FACT*RJ*RJ
  MJ=MJ+RJ
  G1=G1+X2*J*FACT*(.5+(A-MJ)*FLOAT(J))
  IF(N=1131,52,51
  IF(N=1131,52,51
52  BK=G1
  RETURN
  END

```

```

BESK 1
BESK 2
BESK 3
BESK 4
BESK 5
BESK 6
BESK 7
BESK 8
BESK 9
BESK M01
BESK M02
BESK M03
BESK M04
BESK 11
BESK 12
BESK 13
BESK 14
BESK 15
BESK 16
BESK 17
BESK 18
BESK 19
BESK 20
BESK 21
BESK 22
BESK 23
BESK 24
BESK 25
BESK 26
BESK 27
BESK 28
BESK 29
BESK 30
BESK 31
BESK 32
BESK 33
BESK 34
BESK 35
BESK 36
BESK 37
BESK M05
BESK M06
BESK M07
BESK M08
BESK 39
BESK M09
BESK 41
BESK 42
BESK 43
BESK 44
BESK 45
BESK 46
BESK 47
BESK 48
BESK 49
BESK 50
BESK 51
BESK 52
BESK 53
BESK 54
BESK 55
BESK 56
BESK 57
BESK 58
BESK 59
BESK 60
BESK 61
BESK 62
BESK 63
BESK 64
BESK 65
BESK 66
BESK 67
BESK 68
BESK 69
BESK 70
BESK 71
BESK 72
BESK 73
BESK 74

```

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)
  IER=0
  TEST MODULUS
  GEO=1.-AK*AK
  IF(GEO)1,2,3
1  IER=1
  RETURN
  SET RESULT VALUE =OFLOW
2  RES=1.E38
  RETURN
3  GEO=SQRT(GEO)
  ARI=1.
4  AARI=ARI
  TEST=AARI*1.E-4
  ARI=GEO*ARI
  TEST OF ACCURACY
  IF(AARI-GEO-TEST)6,6,5
5  GEO=SQRT(AARI*GEO)
  ARI=0.5*ARI
  GO TO 4
6  RES=9.141993 /ARI
  RETURN
END

```

```

CEL1 1
CEL1 2
CEL1 3
CEL1 4
CEL1 5
CEL1 6
CEL1 7
CEL1 8
CEL1 9
CEL1 10
CEL1 11
CEL1 12
CEL1 13
CEL1 14
CEL1 15
CEL1 16
CEL1 17
CEL1 18
CEL1 19
CEL1 20
CEL1 21
CEL1 22
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:

$$k_c^2 + k^2 = 1, \quad 0 < k_c^2 \leq 1$$

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:

$$a_0 = 1, \quad g_0 = k_c$$

the sequences of numbers  $(a_n)$ ,  $(g_n)$  are generated using for definition:

$$a_n = (a_{n-1} + g_{n-1}), \quad g_n = 2 \sqrt{a_{n-1} g_{n-1}}$$

This iteration process is stopped at the  $N^{\text{th}}$  step, when  $a_N = g_N$ .

Further needed are the sequences

$(A_i)$ ,  $(B_i)$  defined by means of:

$$A_0 = A, \quad B_0 = B$$

$$A_n = B_{n-1} / a_{n-1} + A_{n-1}$$

$$B_n = 2 (B_{n-1} + g_{n-1} \cdot A_{n-1})$$

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.

```

4 RES=1.E38
5 RETURN
5 RES=A
6 RETURN
6 COMPUTE INTEGRAL
6 GEO=SQRT(GEO)
  ARI=1.
  AA=A
  AN=AA*B
  W=B
  W=W+AA*GEO
  W=W*W
  AA=AN
  AARI=ARI
  ARI=GEO+ARI
  AN=W/ARI+AN
  TEST OF ACCURACY
  IF(AARI-GEO-1.E-4*AA*ARI)9,9,8
8 GEO=SQRT(GEO+AARI)
  GEO=GEO+GEO
  GO TO 7
9 RES=-.7853982 *AN/ARI
  RETURN
  END

```

```

CEL2 13
CEL2 14
CEL2 15
CEL2 16
CEL2 17
CEL2 18
CEL2 19
CEL2 20
CEL2 21
CEL2 22
CEL2 23
CEL2 24
CEL2 25
CEL2 26
CEL2 27
CEL2 28
CEL2 29
CEL2 30
CEL2 31
CEL2 32
CEL2 33
CEL2 34
CEL2 35

```

```

SUBROUTINE CEL2(RES,AK,A,B,IER)
  IER=0
  TEST MODULUS
  GEO=1.-AK*AK
  IF(GEO)1,2,6
1 IER=1
  RETURN
2 SET RESULT VALUE = OVERFLOW
3 IF(B)3,3,4
3 RES=1.E38
  RETURN

```

```

CEL2 1
CEL2 2
CEL2 3
CEL2 4
CEL2 5
CEL2 6
CEL2 7
CEL2 8
CEL2 9
CEL2 10
CEL2 11
CEL2 12

```

## 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  $-E_1(-x)$ ,

where

$$E_1(y) = - \int_{-y}^{\infty} \frac{e^{-t}}{t} dt, \quad y > 0$$

( $\int$  denotes Cauchy principal value.)

For  $x = 0$ , a singularity of the function, the program returns  $1.0 \times 10^{38}$ .

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  $T_n^*$

$$E_1(x) = \frac{e^{-x}}{x} \sum_{n=0}^{\infty} A_n T_n^*\left(\frac{4}{x}\right), \quad \text{for } 4 \leq x < \infty$$

---

\*Luke/Wimp, "Jacobi Polynomial expansion of a generalized hypergeometric function over a semi-infinite ray", Math. Comp., Vol. 17, 1963, Iss. 84, p. 400.

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 polynomials to ordinary polynomials finally leads to the approximation:

$$\text{EXPI}(x) = e^{-x} \left(\frac{4}{x}\right) \sum_{v=0}^{\infty} a_v \left(\frac{4}{x}\right)^v \quad \text{for } x \geq 4$$

The coefficients of this approximation given to eight significant digits are:

$$\begin{aligned} a_0 &= 0.24999 \ 999 \\ a_1 &= -0.06249 \ 8588 \\ a_2 &= 0.03120 \ 8561 \\ a_3 &= -0.02295 \ 1979 \\ a_4 &= 0.02041 \ 2099 \\ a_5 &= -0.01755 \ 5779 \\ a_6 &= 0.01172 \ 3273 \\ a_7 &= -0.00493 \ 62007 \\ a_8 &= 0.00094 \ 42761 \ 4 \end{aligned}$$

### 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:

$$\text{EXPI}(x) = -\ln|x| + \sum_{v=0}^{14} b_v x^v$$

with a truncation error E absolutely less than  $3 \times 10^{-8}$ .

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 EXP1

Purpose:

Computes the exponential integral in the range -4 to infinity.

Usage:

CALL EXP1(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( $\text{EXP}(-T)/T$ , 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.

```

C      SUBROUTINE EXP1(RES,X,IER)
C      TEST OF RANGE
C      IER=0
C      IF(X=0) 10,10,20
C      IF(X=4) 55,30,33
C      ARGUMENT IS GREATER THAN 4
C      20 ARG=4./X
C      RES=EXP(-X)*(((((((1.0009442764*ARG-.0049362007)*ARG+.011723273)
C      1 *ARG-.017555779)*ARG+.020412099)*ARG-.022951979)*ARG+.031208561)
C      2 *ARG-.062498588)*ARG+.24999999)*ARG
C      RETURN
C      ARGUMENT IS ABSOLUTELY LESS OR EQUAL 4
C      30 IF(X1 40,50,40
C      400 RES=-ALOG(ABS(X))-(((((((1.0317602E-1)*X-.15798675E-10)*X
C      1.16826592E-9)*X-.21915699E-8)*X+.27615930E-7)*X-.30726221E-6)*X
C      2.30996040E-5)*X-.28337590E-4)*X+.23148392E-3)*X-.0016666906)*X
C      3.010416662)*X-.05555520)*X+.251)*X-1.01)*X-.57721566
C      RETURN
C      50 RES=1.E38
C      RETURN
C      ARGUMENT IS LESS THAN -4.
C      55 IER=1
C      RETURN
C      END
EXP1 1
EXP1 2
EXP1 3
EXP1 4
EXP1 5
EXP1 6
EXP1 7
EXP1 8
EXP1 9
EXP1 10
EXP1 11
EXP1 12
EXP1 13
EXP1 14
EXP1 15
EXP1 16
EXP1 17
EXP1 18
EXP1 19
EXP1 20
EXP1 21
EXP1 22
EXP1 23
EXP1 24

```

## SICI

This subroutine computes the sine and cosine integrals. These integrals are defined as:

$$\text{Si}(x) = \int_0^x \frac{\sin(t)}{t} dt, \quad x \geq 0$$

and

$$\text{Ci}(x) = \int_x^\infty \frac{\cos(t)}{t} dt, \quad x > 0$$

The subroutine SICI calculates both  $\text{Si}(x)$  and  $\text{Ci}(x)$  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:

$$\text{Si}(-x) = -\pi - \text{Si}(x)$$

Real part of

$$\text{Ci}(-x) = \text{Ci}(x), \quad x > 0 \text{ (see discussion of EXPI).}$$

For  $x = 0$ , a singularity of  $\text{Ci}(x)$ , the routine returns  $-1.0 \times 10^{38}$ .

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:

$$\text{Si}(x) + i \text{Ci}(x) = \frac{\pi}{2} + ie^{ix} \Psi(1, 1; -ix).$$

Setting:

$$ix \Psi(1, 1; ix) = \sum_{n=0}^{\infty} (A_n + i B_n) T_n^* \left( \frac{4}{x} \right)$$

\*Luke/Wimp, "Jacobi Polynomial expansion of a generalized hypergeometric function over a semi-infinite ray", Math. Comp. Vol. 17, 1963, Iss. 84, p. 402.

we get the expansions:

$$\text{Si}(x) = \sum_{n=0}^{\infty} \left( \frac{A_n \cdot \cos x}{x} + \frac{B_n \cdot \sin x}{x} \right) T_n^* \left( \frac{4}{x} \right)$$

$$\text{Ci}(x) = \sum_{n=0}^{\infty} \left( \frac{B_n \cdot \cos x}{x} - \frac{A_n \cdot \sin x}{x} \right) T_n^* \left( \frac{4}{x} \right)$$

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:

$$|E(x)| < \frac{1}{x} \cdot 2.3 \cdot 10^{-8}$$

Transformation of the shifted Chebyshev polynomials to ordinary polynomials finally leads to the approximations:

$$\text{Si}(x) = -\left(\frac{4}{x}\right) \cdot (\cos x \cdot V(x) + \sin x \cdot U(x))$$

$$\text{Ci}(x) = \left(\frac{4}{x}\right) \cdot (\sin x \cdot V(x) - \cos x \cdot U(x)),$$

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      |

$a_3 = -0.00037 \ 64000 \ 3$   
 $b_3 = -0.02314 \ 6168$   
 $a_4 = 0.02601 \ 2930$   
 $b_4 = -0.00333 \ 25186$   
 $a_5 = -0.00794 \ 55563$   
 $b_5 = 0.04987 \ 7159$   
 $a_6 = -0.04400 \ 4155$   
 $b_6 = -0.07261 \ 6418$   
 $a_7 = 0.07902 \ 0335$   
 $b_7 = 0.05515 \ 0700$   
 $a_8 = -0.06537 \ 2834$   
 $b_8 = -0.02279 \ 1426$   
 $a_9 = 0.02819 \ 1786$   
 $b_9 = 0.00404 \ 80690$   
 $a_{10} = -0.00510 \ 86993$

## 2. Approximation in the range $|x| \leq 4$ .

A polynomial approximation for Si(x) is obtained by means of telescoping of the Taylor series:

$$\begin{aligned}
 \text{Si}(x) &= -\frac{\pi}{2} + \int_0^x \frac{\sin t}{t} dt \\
 &= -\frac{\pi}{2} + x \cdot \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n+1) \cdot (2n+1)!}
 \end{aligned}$$

This results in the approximation:

$$\text{Si}(x) = -\frac{\pi}{2} + x \cdot \sum_{n=0}^6 a_n (x^2)^n,$$

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:

$$\text{Ci}(x) = C - \ln(x) = \sum_{n=1}^{\infty} \frac{(-1)^n x^{2n}}{2n \cdot (2n)!}$$

This results in the approximation:

$$\text{Ci}(x) = C + \ln|x| - x^2 \cdot \sum_{n=0}^5 b_n (x^2)^n,$$

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.

Luke and Wimp, 'Polynomial Approximations to Integral Transforms', *Mathematical Tables and Other Aids to Computation*, Vol. 15, 1961, Issue 74, pp. 174-178.

|      |    |
|------|----|
| SICI | 1  |
| SICI | 2  |
| SICI | 3  |
| SICI | 4  |
| SICI | 5  |
| SICI | 6  |
| SICI | 7  |
| SICI | 8  |
| SICI | 9  |
| SICI | 10 |
| SICI | 11 |
| SICI | 12 |
| SICI | 13 |
| SICI | 14 |
| SICI | 15 |
| SICI | 16 |
| SICI | 17 |
| SICI | 18 |
| SICI | 19 |
| SICI | 20 |
| SICI | 21 |
| SICI | 22 |
| SICI | 23 |
| SICI | 24 |
| SICI | 25 |
| SICI | 26 |
| SICI | 27 |
| SICI | 28 |
| SICI | 29 |
| SICI | 30 |
| SICI | 31 |
| SICI | 32 |
| SICI | 33 |

\*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:

$$C(x) = \frac{1}{2} + \sqrt{\frac{4}{x}} (\sin(x) \cdot P(x) + \cos(x) \cdot Q(x))$$

$$S(x) = \frac{1}{2} + \sqrt{\frac{4}{x}} (-\cos(x) \cdot P(x) + \sin(x) \cdot Q(x))$$

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 $0 \leq x \leq 4$ .

Approximations for  $C(x)$  and  $S(x)$  in the range  $0 \leq x \leq 4$  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 CS(C,S,X)
Z=ABS(X)
2 IF(Z<4.) 3,3,4
C X IS NOT GREATER THAN 4
3 C=SQRT(Z)
S=Z*C
Z=Z**2
C=C*(((11.50998348E-10)*Z-1.0140729E-7)*Z+.11605286E-5)*Z
1 -.85224622E-4)*Z+.3693898E-2)*Z-.079788405)*Z+.797884551
S=S*(((11.-.66777647E-9)*Z+.11275331E-6)*Z-1.0525853E-4)*Z
1 +.60435371E-3)*Z-.18997110E-1)*Z+.26596149)
RETURN
C X IS GREATER THAN 4
4 D=COS(Z)
S=SIN(Z)
Z=4./Z
A=(((111.87682593E-3)*Z-.41692894E-2)*Z+.79709430E-2)*Z-
1.67928011E-2)*Z-.30953612E-3)*Z+.59721508E-2)*Z-.16064281E-4)*Z-
2.024933215)*Z-.44440900E-8
B=(((111.-.66339250E-3)*Z+.34014090E-2)*Z-.72716901E-7)*Z+
1.74282459E-2)*Z-.40271450E-3)*Z-.93149105E-2)*Z-.12079986E-5)*Z+
2.1994711
Z=SQRT(Z)
C=.5*Z*(D*A+S*B)
S=.5*Z*(S*A-D*B)
RETURN
END
CS 1
CS 2
CS 3
CS 4
CS 5
CS 6
CS 7
CS 8
CS 9
CS 10
CS 11
CS 12
CS 13
CS 14
CS 15
CS 16
CS 17
CS 18
CS 19
CS 20
CS 21
CS 22
CS 23
CS 24
CS 25
CS 26
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:

$C(X) = \text{integral } (\cos(T)/\sqrt{2\pi T}) \text{ summed over } T \text{ from } 0 \text{ to } X.$

$S(X) = \text{integral } (\sin(T)/\sqrt{2\pi T}) \text{ summed over } T \text{ from } 0 \text{ to } X.$

##### 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.

## Mathematics - Linear Equations

### 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)
  DIMENSION A(1),B(1)
  FORWARD SOLUTION
  TOL=0.0
  KS=0
  JJ=N
  DO 65 J=1,N
    JY=J+1
    JJ=JJ+N+1
    BIGA=0
    IT=JJ-J
    DO 30 I=J,N

```

```

C      SEARCH FOR MAXIMUM COEFFICIENT IN COLUMN
  IJ=IT+1
  IF (ABS(BIGA)-ABS(A(I,J))) 20,30,30
20  BIGA=A(I,J)
  IMAX=I
30  CONTINUE
C      TEST FOR PIVOT LESS THAN TOLERANCE (SINGULAR MATRIX)
  IF (ABS(BIGA)-TOL) 35,35,40
35  KS=1
  RETURN
C      INTERCHANGE ROWS IF NECESSARY
40  II=IMAX-(J-2)
  IT=IMAX-J
  DO 50 K=J,N
    I1=I+K
    I2=II+K
    SAVE=A(I1)
    A(I1)=A(I2)
    A(I2)=SAVE
C      DIVIDE EQUATION BY LEADING COEFFICIENT
50  A(I1)=A(I1)/BIGA
  SAVE=B(IMAX)
  B(IMAX)=B(J)
  B(J)=SAVE/BIGA
C      ELIMINATE NEXT VARIABLE
  IF (J=N) 55,70,55
55  IQS=N-(J-1)
  DO 65 IX=JY,N
    IJ=IQS+IX
    IT=J-IX
    DO 60 JX=JY,N
      IXJN=N-(IX-1)+IX
      JXN=IXJN+IT
60  A(IXJN)=A(IXJN)-(A(IXJ)*A(JJX))
65  B(IX)=B(IX)-(B(J)*A(IXJ))
C      BACK SOLUTION
70  NY=N-1
  IT=NY
  DO 80 J=1,NY
    IA=IT-J
    IB=NY-IA
    IC=
    DO 80 K=1,J
      B(IB)=B(IB)-A(IA)*B(IC)
    IA=IA-N
    IC=IC-1
  RETURN
END

```

## Mathematics - Roots of Nonlinear Equations

### 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:

$$x_2 = x_1 + \frac{x_1 - x_0}{\frac{x_0 - y_0}{x_1 - y_1} - 1}$$

$$\text{and } y_2 = f(x_2)$$

The next step is done by starting at  $(x_2, y_2)$  and setting:

$$x_3 = x_2 + \frac{x_2 - x_1}{\frac{x_1 - y_1}{x_2 - y_2} - 1}$$

$$y_3 = f(x_3)$$

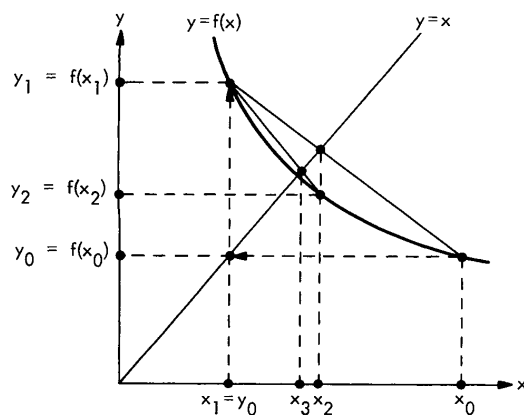


Figure 8. Wegstein's iterative method

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:

$$\left. \begin{aligned} x_{i+1} &= x_i + \frac{x_i - x_{i-1}}{\frac{x_{i-1} - y_{i-1}}{x_i - y_i} - 1} \\ y_{i+1} &= f(x_{i+1}) \end{aligned} \right\} (i = 1, 2, \dots) \quad (1)$$

Each step requires one evaluation of  $f(x)$ .

This iterative procedure is terminated if the following two conditions are satisfied:

$$\left. \begin{aligned} \delta_1 &\leq \quad \text{and} \quad \delta_2 \leq 10 \cdot \epsilon \\ \text{with } \delta_1 &= \begin{cases} \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| & \text{if } |x_{i+1}| > 1 \\ |x_{i+1} - x_i| & \text{if } |x_{i+1}| \leq 1 \end{cases} \\ \delta_2 &= \begin{cases} \left| \frac{x_{i+1} - y_{i+1}}{x_{i+1}} \right| & \text{if } |x_{i+1}| > 1 \\ |x_{i+1} - y_{i+1}| & \text{if } |x_{i+1}| \leq 1 \end{cases} \end{aligned} \right\} (2)$$

and tolerance  $\epsilon$  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  $\epsilon$  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  $\xi$  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
  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
  START ITERATION LOOP                           RTWI 10
  DO 6 J=1,IEND                                  RTWI 11
  IF(VAL)1,7,1                                    RTWI 12
  IF(VAL)1,7,1                                    RTWI 13
  1 B=B/VAL-1,                                     RTWI 14
  IF(B)2,8,2                                       RTWI 15
  IF(B)2,8,2                                       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
  TEST ON SATISFACTORY ACCURACY                  RTWI 22
  TOL=EPS                                          RTWI 23
  D=ABS(X)                                         RTWI 24
  IF(D-1.)4,4,3                                    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
  END OF ITERATION LOOP                           RTWI 30
  NO CONVERGENCE AFTER IEND ITERATION STEPS. ERROR RETURN. RTWI 31
  IER=1                                           RTWI 32
  7 RETURN                                         RTWI 33
  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  $f(x_{li}) \cdot f(x_{ri}) \leq 0$ .

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  $x_l \dots x_r$  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

$2 f(x_m) [f(x_m) - f(x_l)] - f(x_r) [f(x_r) - f(x_l)] \geq 0$  (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:

$$\Delta x = f(x_l) \frac{x_m - x_l}{f(x_m) - f(x_l)} \left\{ 1 + f(x_m) \frac{f(x_r) - 2f(x_m) + f(x_l)}{[f(x_r) - f(x_m)][f(x_r) - f(x_l)]} \right\} \quad (2)$$

$$\text{and } x = x_l - \Delta x$$

and x is sure to be situated between  $x_l$  and  $x_m$

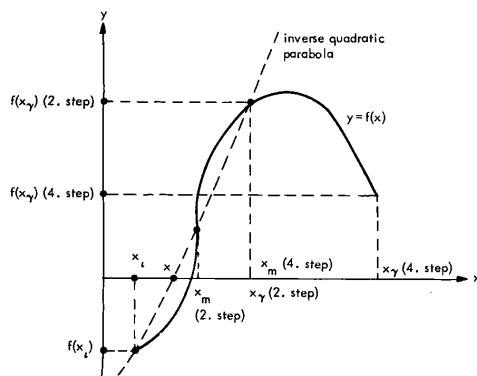


Figure 9. Mueller's iterative method

Now, for the next iteration step,  $x$  becomes  $x_1$  and  $x_m$  becomes  $x_r$  if  $f(x) \cdot f(x_1) > 0$ , or  $x$  becomes  $x_r$  if  $f(x) \cdot f(x_1) < 0$ .

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  $x_{li} \dots x_{ri}$ . Each iteration step requires two evaluations of  $f(x)$ .

This iterative procedure is terminated if either the two conditions (checked in bisection loop)

$$\left. \begin{aligned} |x_r - x_l| &\leq \epsilon \cdot \max(1, |x_r|) \\ |f(x_r) - f(x_l)| &\leq 100 \cdot \epsilon \end{aligned} \right\} \quad (3)$$

or the two conditions (checked after inverse parabolic interpolation)

$$\left. \begin{aligned} |\Delta x| &\leq \epsilon \cdot \max(1, |x|) \\ |f(x)| &\leq 100 \cdot \epsilon \end{aligned} \right\} \quad (4)$$

are satisfied, where tolerance  $\epsilon$  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  $x_{li} \dots x_{ri}$  is too long.
3. The tolerance  $\epsilon$  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  $FCT(XLI) \cdot FCT(XRI)$  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.

```

C SUBROUTINE RTNI(X,F,FCT,XL1,XR1,EPS,IEND,IER)
C PREPARE ITERATION
IER=0
XL=XL1
XR=XR1
X=XL
TOL=X
F=FCT(TOL)
IF(F)1,16,1
1 FL=F
X=XR
TOL=X
F=FCT(TOL)
IF(F)2,16,2
2 FR=F
IF(SIGN(1,FL)+SIGN(1,FR))25,3,25
C BASIC ASSUMPTION FL*FR LESS THAN 0 IS SATISFIED.
C GENERATE TOLERANCE FOR FUNCTION VALUES.
3 I=0
TOL=F*100.*EPS
C START ITERATION LOOP
4 I=1
C START BISECTION LOOP
DO 13 K=1,IEND
X=.5*(XL+XR)
TOL=X
F=FCT(TOL)
IF(F)5,16,5
5 IF(SIGN(1,F)+SIGN(1,FR))7,6,7
C INTERCHANGE XL AND XR IN ORDER TO GET THE SAME SIGN IN F AND FR
6 TOL=XL
XL=XR
XR=TOL
TOL=FL
FL=FR
FR=TOL
7 TOL=FL
A=F*TOL
A=A+FR
IF(A*(FR-FL))8,9,9
8 IF(I-IEND)17,17,9
9 XR=X
FR=F
C TEST ON SATISFACTORY ACCURACY IN BISECTION LOOP
TOL=EPS
A=ABS(XR)
IF(A-1.)11,11,10
10 TOL=TOL*A
11 IF(ABS(XR-XL)-TOL)12,12,13
12 IF(ABS(FR-FL)-TOL)14,14,13
13 CONTINUE
C END OF BISECTION LOOP
C NO CONVERGENCE AFTER IEND ITERATION STEPS FOLLOWED BY IEND.
C SUCCESSIVE STEPS OF BISECTION OR STEADILY INCREASING FUNCTION
C VALUES AT RIGHT BOUNDS. ERROR RETURN.
IER=1
14 IF(ABS(FR)-ABS(FL))16,16,15
15 X=XL
F=FL
16 RETURN
C COMPUTATION OF ITERATED X-VALUE BY INVERSE PARABOLIC INTERPOLATION
17 A=FR-F
DX=(X-XL)*FL*(1.+F*(A-TOL)/(A*(FR-FL)))/TOL
X=X+X
F=F
X=XL+DX
TOL=X
F=FCT(TOL)
IF(F)18,16,18
C TEST ON SATISFACTORY ACCURACY IN ITERATION LOOP
18 TOL=EPS
A=ABS(X)
IF(A-1.)20,20,19
19 TOL=TOL*A
20 IF(ABS(DX)-TOL)21,21,22
21 IF(ABS(F)-TOL)16,16,22
C PREPARATION OF NEXT BISECTION LOOP
22 IF(SIGN(1,F)+SIGN(1,FL))24,23,24
23 XR=X
FR=F
GO TO 4
24 XL=X
FL=F
XR=X
FR=F
GO TO 4
C END OF ITERATION LOOP
C ERROR RETURN IN CASE OF WRONG INPUT DATA
25 IER=2
RETURN
END

```

## 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:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad (i = 0, 1, 2, \dots) \quad (1)$$

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:

$$\delta \leq \epsilon \text{ and } |f(x_{i+1})| \leq 100 \cdot \epsilon$$

$$\text{with } \delta = \begin{cases} \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| & \text{in case of } |x_{i+1}| > 1 \\ |x_{i+1} - x_i| & \text{in case of } |x_{i+1}| \leq 1 \end{cases} \quad (2)$$

and tolerance  $\epsilon$  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  $\epsilon$  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.

```

C      SUBROUTINE RTNI(X,F,DERF,FCT,XST,EPS,IEND,IER)      RTNI 1
C      PREPARE ITERATION                                RTNI 2
C      IER=0                                              RTNI 3
C      X=XST                                             RTNI 4
C      TOL=X                                              RTNI 5
C      CALL FCT(TOL,F,DERF)                               RTNI 6
C      TOLF=100.*EPS                                     RTNI 7
C      START ITERATION LOOP                             RTNI 8
C      DO 6 I=1,IEND                                     RTNI 9
C      IF(I11,7,1)                                       RTNI 10
C      EQUATION IS NOT SATISFIED BY X                   RTNI 11
C      1 IF(DERF)2,8,2                                  RTNI 12
C      ITERATION IS POSSIBLE                            RTNI 13
C      2 DX=F/DERF                                       RTNI 14
C      X=X+DX                                            RTNI 15
C      TOL=X                                             RTNI 16
C      CALL FCT(TOL,F,DERF)                               RTNI 17
C      TEST ON SATISFACTORY ACCURACY                     RTNI 18
C      TOL=EPS                                           RTNI 19
C      A=ABS(X)                                           RTNI 20
C      IF(A-1,34,4,3)                                    RTNI 21
C      3 TOL=TOL*A                                       RTNI 22
C      4 IF(ABS(DX)-TOL)5,5,6                             RTNI 23
C      5 IF(ABS(F)-TOLF)7,7,6                             RTNI 24
C      6 CONTINUE                                       RTNI 25
C      END OF ITERATION LOOP                             RTNI 26
C      NO CONVERGENCE AFTER IEND ITERATION STEPS- ERROR RETURN- RTNI 27
C      IER=1                                             RTNI 28
C      7 RETURN                                           RTNI 29
C      ERROR RETURN IN CASE OF ZERO DIVISOR             RTNI 30
C      8 IER=2                                           RTNI 31
C      RETURN                                           RTNI 32
C      END                                              RTNI 33

```

## Mathematics - Roots of Polynomial

### POLRT

This subroutine computes the real and complex roots of a real polynomial.

Given a polynomial

$$f(z) = \sum_{n=0}^N a_n z^n \quad (1)$$

let

$Z = X + iY$  be a starting value for a root of  $f(z)$ .

Then:

$$Z^n = (X + iY)^n. \quad (2)$$

Define  $X_n$  as real terms of expanded equation (2).

Define  $Y_n$  as imaginary terms of expanded equation (2).

Then for:

$$n = 0$$

$$X_0 = 1.0$$

$$Y_0 = 0.0$$

$$n > 0$$

$$X_n = X \cdot X_{n-1} - Y \cdot Y_{n-1} \quad (3)$$

$$Y_n = X \cdot Y_{n-1} + Y \cdot X_{n-1} \quad (4)$$

Let  $U$  be the real terms of (1).

$V$  be the imaginary terms of (1).

Then:

$$U = \sum_{n=0}^N a_n X_n \quad (5)$$

$$V = \sum_{n=0}^N a_n Y_n \quad (6)$$

or

$$U = a_0 + \sum_{n=1}^N a_n X_n \quad (7)$$

$$V = \sum_{n=1}^N a_n Y_n \quad (8)$$

$$\frac{\partial U}{\partial X} = \sum_{n=1}^N n \cdot X_{n-1} \cdot a_n \quad (9)$$

$$\frac{\partial U}{\partial Y} = - \sum_{n=1}^N n Y_{n-1} a_n \quad (10)$$

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  $Y_{n-1}$ .

$$\Delta X = \left( V \frac{\partial U}{\partial Y} - U \frac{\partial U}{\partial X} \right) / \left[ \left( \frac{\partial U}{\partial X} \right)^2 + \left( \frac{\partial U}{\partial Y} \right)^2 \right] \quad (11)$$

Thus, for the next iteration:

$$Y' = Y + \Delta Y$$

**Purpose:**

### Usage:

Description of parameters:

IER - Error code where:

Remarks:

Subroutines and function subprograms required:

None.

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.

Mathematics — Special Operations and Functions 121

## Mathematics - Polynomial Operations

### 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)
C
  DIMENSION Z(1), X(1), Y(1)
  TEST DIMENSIONS OF SUBRANSOS
  NDIM=IDIMX
  IF (IDIMX-IDIMY) 10,20,20
10 NDIM=IDIMY
20 IF(NDIM) 90,90,30
30 DO 80 I=1,NDIM
  IF(I-IDIMX) 40,40,60
  IF(I-IDIMY) 50,50,70
50 Z(I)=X(I)+Y(I)
  GO TO 80
60 Z(I)=Y(I)
  GO TO 80
70 Z(I)=X(I)
80 CONTINUE
90 IDIMZ=NDIM
RETURN
END
```

### 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)
C
  DIMENSION Z(1), X(1), Y(1)
  TEST DIMENSIONS OF SUBRANSOS
  NDIM=IDIMX
  IF (IDIMX-IDIMY) 10,20,20
10 NDIM=IDIMY
20 IF(NDIM) 90,90,30
30 DO 80 I=1,NDIM
  IF(I-IDIMX) 40,40,60
  IF(I-IDIMY) 50,50,70
50 Z(I)=FACT*Y(I)+X(I)
  GO TO 80
60 Z(I)=X(I)
  GO TO 80
70 Z(I)=Y(I)
80 CONTINUE
90 IDIMZ=NDIM
RETURN
END
```

### 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)
  DIMENSION X(1), Y(1)
  IDIMY = IDIMX
  IF (IDIMX) 30, 30, 10
10 DO 20 I=1, IDIMX
  20 Y(I) = X(I)
  30 RETURN
  END
PCLA 1
PCLA 2
PCLA 3
PCLA 4
PCLA 5
PCLA 6
PCLA 7
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)
  DIMENSION Z(1), X(1), Y(1)
  C TEST DIMENSIONS OF SUMMANDS
  NDIM = IDIMX
  IF (IDIMX - IDIMY) 10, 20, 20
10 NDIM = IDIMY
20 IF (NDIM) 90, 90, 30
30 DO 80 I=1, NDIM
  IF (I - IDIMX) 40, 40, 60
40 IF (I - IDIMY) 50, 50, 70
50 Z(I) = X(I) - Y(I)
  GO TO 80
60 Z(I) = -Y(I)
  GO TO 80
70 Z(I) = X(I)
80 CONTINUE
90 IDIMZ = NDIM
  RETURN
  END
PSUB 1
PSUB 2
PSUB 3
PSUB 4
PSUB 5
PSUB 6
PSUB 7
PSUB 8
PSUB 9
PSUB 10
PSUB 11
PSUB 12
PSUB 13
PSUB 14
PSUB 15
PSUB 16
PSUB 17
PSUB 18
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)
  DIMENSION Z(1), X(1), Y(1)
  IF (IDIMX + IDIMY - 1) 10, 20
10 IDIMZ = 0
  GO TO 50
20 IDIMZ = IDIMX + IDIMY - 1
  DO 30 I=1, IDIMZ
  30 Z(I) = 0
  DO 40 J=1, IDIMX
  DO 40 K=1, IDIMY
  40 Z(I+J+K-1) = Z(I+J+K-1) + X(J)*Y(K)
  50 RETURN
  END
PMPY 1
PMPY 2
PMPY 3
PMPY 4
PMPY 5
PMPY 6
PMPY 7
PMPY 8
PMPY 9
PMPY 10
PMPY 11
PMPY 12
PMPY 13
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)
  DIMENSION P(1), X(1), Y(1)
  CALL PNORM(Y, IDIMY, TOL)
  IF(IDIMY) 50, 50, 10
  10 IDIMP=IDIMX-IDIMY+1
  IF(IDIMP) 20, 30, 50
  C DEGREE OF DIVISOR WAS GREATER THAN DEGREE OF DIVIDEND
  20 IDIMP=0
  30 IER=0
  40 RETURN
  C Y IS ZERO POLYNOMIAL
  50 IER=1
  GO TO 40
  C START REDUCTION
  60 IDIMX=IDIMY-1
  I=IDIMP
  70 II=I-IDIMX
  P(II)=X(II)/Y(IDIMY)
  C SUBTRACT MULTIPLE OF DIVISOR
  DO 80 K=1, IDIMX
    J=K-1+I
    X(J)=X(J)-P(II)*Y(K)
  80 CONTINUE
  I=I-1
  IF(I) 90, 90, 70
  C NORMALIZE REMAINDER POLYNOMIAL
  90 CALL PNORM(X, IDIMX, TOL)
  GO TO 30
END
```

## 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)
  DIMENSION X(1)
  A=0.
  B=0.
  J=IDIMX
  1 IF(J) 3, 2
  2 Z=P*A+X(J)
  B=Q*A+X(J)
  A=Z
  J=J-1
  GO TO 1
  3 RETURN
END
```

PQSD 1  
PQSD 2  
PQSD 3  
PQSD 4  
PQSD 5  
PQSD 6  
PQSD 7  
PQSD 8  
PQSD 9  
PQSD 10  
PQSD 11  
PQSD 12  
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) 1,3,1                                           PVSUB 4
1 IDIMZ=0                                                       PVSUB 5
2 RETURN                                                       PVSUB 6
3 IDIMZ=1                                                       PVSUB 7
IF (IDIMY>IDIMX-IDIMY) 2,2,4                                   PVSUB 8
4 I=1                                                           PVSUB 9
WORK1(1)=1.                                                    PVSUB 10
DO 5 I=2, IDIMX                                                 PVSUB 11
CALL PMPY(WORK2, I, Y, IDIMY, WORK1, I)                      PVSUB 12
CALL PCLA(WORK1, I, WORK2, I)                                   PVSUB 13
CALL PCLA(WORK1, I, WORK2, I)                                   PVSUB 14
FACT=X(I)                                                       PVSUB 15
CALL PADDM(Z, IDIMZ, I, IDIMZ, FACT, WORK1, I)               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) 4,4,1                    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)
DIMENSION X(1)
P=ARGUM*ARGUM
Q=-ARGUM*ARGUM
CALL PQSD (DVAL,POLY,P,Q,X,IDIMX)
POLY=ARGUM*DVAL+POLY
RETURN
END
```

PILD 1  
PILD 2  
PILD 3  
PILD 4  
PILD 5  
PILD 6  
PILD 7  
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)
DIMENSION X(1), Y(1)
TEST OF DIMENSION
IF (IDIMX-1) 3, 3, 1
1 IDIMY=IDIMX-1
EXPT=0.
DO 2 I=1, IDIMY
EXPT=EXPT+1.
2 Y(I)=X(I+1)*EXPT
GO TO 4
3 IDIMY=0
4 RETURN
END
```

PDER 1  
PDER 2  
PDER 3  
PDER 4  
PDER 5  
PDER 6  
PDER 7  
PDER 8  
PDER 9  
PDER 10  
PDER 11  
PDER 12  
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)
DIMENSION X(1), Y(1)
IDIMY=IDIMX+1
Y(1)=0.
IF (IDIMX) 1, 1, 2
1 RETURN
2 EXPT=1.
DO 3 I=2, IDIMY
Y(I)=X(I-1)/EXPT
3 EXPT=EXPT+1.
GO TO 1
END
```

PINT 1  
PINT 2  
PINT 3  
PINT 4  
PINT 5  
PINT 6  
PINT 7  
PINT 8  
PINT 9  
PINT 10  
PINT 11  
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.

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.

#### 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 PNORM(X, IDIMX, EPS)
  DIMENSION X(1)
  1 IF (IDIMX) 6, 4, 2
  2 IF (ABS(X(IDIMX)) - EPS) 1, 3, 4
  3 IDIMX = IDIMX - 1
  GO TO 1
  4 RETURN
END
PNORM 1
PNORM 2
PNORM 3
PNORM 4
PNORM 5
PNORM 6
PNORM 7
PNORM 8

```

```

SUBROUTINE PGCD(X, IDIMX, Y, IDIMY, WORK, EPS, IER)
  DIMENSION X(1), Y(1), WORK(1)
  C DIMENSION REQUIRED FOR VECTOR NAMED WORK IS IDIMX-IDIMY+1
  1 CALL PDIV(WORK, NDIM, X, IDIMX, Y, IDIMY, EPS, IER)
  IF (IER) 5, 2, 5
  2 IF (IDIMX) 5, 5, 3
  C INTERCHANGE X AND Y
  3 DO 4 J=1, IDIMY
    WORK(J) = X(J)
    X(J) = Y(J)
  4 Y(J) = WORK(J)
  NDIM = IDIMX
  IDIMX = IDIMY
  IDIMY = NDIM
  GO TO 1
  5 RETURN
END
PGCD 1
PGCD 2
PGCD 3
PGCD 4
PGCD 5
PGCD 6
PGCD 7
PGCD 8
PGCD 9
PGCD 10
PGCD 11
PGCD 12
PGCD 13
PGCD 14
PGCD 15
PGCD 16
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.

## 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.

|                                                                                                                                                                                                                                                                                                                                                   |                                                                                    |                                         |                                                                 |                                                                                   |                                         |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|-----------------------------------------|-----------------------------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------------|
| 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. |                                                                                    |                                         | <u>Name</u>                                                     | <u>Function</u>                                                                   | <u>Storage<br/>Required<br/>(Words)</u> |
|                                                                                                                                                                                                                                                                                                                                                   |                                                                                    |                                         | <u>Multiple Linear Regression</u>                               |                                                                                   |                                         |
|                                                                                                                                                                                                                                                                                                                                                   |                                                                                    |                                         | (Usually requires CORRE, ORDER, MINV, MULTR in sequence)        |                                                                                   |                                         |
| <u>Name</u>                                                                                                                                                                                                                                                                                                                                       | <u>Function</u>                                                                    | <u>Storage<br/>Required<br/>(Words)</u> | ORDER                                                           | rearrangement of intercorrelations                                                | 206                                     |
| STATISTICS                                                                                                                                                                                                                                                                                                                                        |                                                                                    |                                         | MULTR                                                           | multiple regression and correlation                                               | 518                                     |
| <u>Data Screening</u>                                                                                                                                                                                                                                                                                                                             |                                                                                    |                                         |                                                                 |                                                                                   |                                         |
| TALLY                                                                                                                                                                                                                                                                                                                                             | totals, means, standard deviations, minimums, maximums                             | 356                                     | <u>Polynomial Regression</u>                                    |                                                                                   |                                         |
|                                                                                                                                                                                                                                                                                                                                                   |                                                                                    |                                         | (Usually requires GDATA, ORDER, MINV, MULTR in sequence)        |                                                                                   |                                         |
| BOUND                                                                                                                                                                                                                                                                                                                                             | selection of observations within bounds                                            | 206                                     | GDATA                                                           | data generation                                                                   | 668                                     |
| SUBST                                                                                                                                                                                                                                                                                                                                             | subset selection from observation matrix (needs user's Boolean subroutine)         | 224                                     | <u>Canonical Correlation</u>                                    |                                                                                   |                                         |
|                                                                                                                                                                                                                                                                                                                                                   |                                                                                    |                                         | (Usually requires CORRE, CANOR, MINV, NROOT, EIGEN in sequence) |                                                                                   |                                         |
| ABSNT                                                                                                                                                                                                                                                                                                                                             | detection of missing data                                                          | 94                                      | CANOR                                                           | canonical correlation (CANOR calls MINV and NROOT)                                | 1132                                    |
| TAB1                                                                                                                                                                                                                                                                                                                                              | tabulation of data (1 variable)                                                    | 612                                     | NROOT                                                           | eigenvalues and eigenvectors of a special nonsymmetric matrix (NROOT calls EIGEN) | 752                                     |
| TAB2                                                                                                                                                                                                                                                                                                                                              | tabulation of data (2 variables)                                                   | 1140                                    |                                                                 |                                                                                   |                                         |
| SUBMX                                                                                                                                                                                                                                                                                                                                             | build subset matrix                                                                | 134                                     |                                                                 |                                                                                   |                                         |
| <u>Elementary Statistics</u>                                                                                                                                                                                                                                                                                                                      |                                                                                    |                                         |                                                                 |                                                                                   |                                         |
| MOMEN                                                                                                                                                                                                                                                                                                                                             | first four moments                                                                 | 404                                     | <u>Analysis of Variance</u>                                     |                                                                                   |                                         |
| TTSTT                                                                                                                                                                                                                                                                                                                                             | tests on population means                                                          | 538                                     | (Usually requires AVDAT, AVCAL, MEANQ in sequence)              |                                                                                   |                                         |
| <u>Correlation</u>                                                                                                                                                                                                                                                                                                                                |                                                                                    |                                         | AVDAT                                                           | data storage allocation                                                           | 326                                     |
| CORRE                                                                                                                                                                                                                                                                                                                                             | means, standard deviations, and correlations (needs user's subroutine to get data) | 1164                                    | AVCAL                                                           | Σ and Δ operation                                                                 | 268                                     |
|                                                                                                                                                                                                                                                                                                                                                   |                                                                                    |                                         | MEANQ                                                           | mean square operation                                                             | 560                                     |

| <u>Name</u>                                                     | <u>Function</u>                                          | <u>Storage<br/>Required<br/>(Words)</u> | <u>Name</u>                      | <u>Function</u>                                               | <u>Storage<br/>Required<br/>(Words)</u> |
|-----------------------------------------------------------------|----------------------------------------------------------|-----------------------------------------|----------------------------------|---------------------------------------------------------------|-----------------------------------------|
| <u>Discriminant Analysis</u>                                    |                                                          |                                         | KRANK                            | Kendall rank correlation (KRANK calls RANK and TIE)           | 524                                     |
| (Usually requires DMATX, MINV, DISCR in sequence)               |                                                          |                                         | WTEST                            | Kendall coefficient of concordance (WTEST calls RANK and TIE) | 498                                     |
| DMATX                                                           | means and dispersion matrix                              | 422                                     | RANK                             | rank observations                                             | 216                                     |
| DISCR                                                           | discriminant functions                                   | 980                                     | TIE                              | calculation of ties in ranked observations                    | 196                                     |
| <u>Factor Analysis</u>                                          |                                                          |                                         | <u>Random Number Generators</u>  |                                                               |                                         |
| (Usually requires CORRE, EIGEN, TRACE, LOAD, VARMX in sequence) |                                                          |                                         | RANDU                            | uniform random numbers                                        | 52                                      |
| TRACE                                                           | cumulative percentage of eigenvalues                     | 160                                     | GAUSS                            | normal random numbers (GAUSS calls RANDU)                     | 68                                      |
| LOAD                                                            | factor loading                                           | 98                                      | <u>MATHEMATICS</u>               |                                                               |                                         |
| VARMX                                                           | varimax rotation                                         | 1186                                    | <u>Special Matrix Operations</u> |                                                               |                                         |
| <u>Time Series</u>                                              |                                                          |                                         | MINV                             | matrix inversion                                              | 784                                     |
| AUTO                                                            | autocovariances                                          | 180                                     | EIGEN                            | eigenvalues and eigenvectors of a real, symmetric matrix      | 1058                                    |
| CROSS                                                           | crosscovariances                                         | 248                                     | <u>Matrices</u>                  |                                                               |                                         |
| SMO                                                             | application of filter coefficients (weights)             | 166                                     | SIMQ                             | solution of simultaneous linear, algebraic equations          | 540                                     |
| EXSMO                                                           | triple exponential smoothing                             | 274                                     | GMADD                            | add two general matrices                                      | 52                                      |
| <u>Nonparametric Statistics</u>                                 |                                                          |                                         | GMSUB                            | subtract two general matrices                                 | 52                                      |
| CHISQ                                                           | $\chi^2$ test for a contingency table                    | 490                                     | GMPRD                            | product of two general matrices                               | 156                                     |
| UTEST                                                           | Mann-Whitney U-test (UTEST calls RANK and TIE)           | 242                                     | GMTRA                            | transpose of a general matrix                                 | 88                                      |
| TWOAV                                                           | Friedman two-way analysis of variance (TWOAV calls RANK) | 324                                     | GTPRD                            | transpose product of two general matrices                     | 152                                     |
| QTEST                                                           | Cochran Q-test                                           | 232                                     | MADD                             | add two matrices (calls LOC)                                  | 226                                     |
| SRANK                                                           | Spearman rank correlation (SRANK calls RANK and TIE)     | 378                                     |                                  |                                                               |                                         |

| <u>Name</u> | <u>Function</u>                                                           | <u>Storage<br/>Required<br/>(Words)</u> | <u>Name</u> | <u>Function</u>                                          | <u>Storage<br/>Required<br/>(Words)</u> |
|-------------|---------------------------------------------------------------------------|-----------------------------------------|-------------|----------------------------------------------------------|-----------------------------------------|
| MSUB        | subtract two matrices<br>(calls LOC)                                      | 226                                     | RTAB        | tabulate the rows of a<br>matrix (calls LOC, RADD)       | 198                                     |
| MPRD        | matrix product (row<br>into column) (calls LOC)                           | 230                                     | CTAB        | tabulate the columns of<br>a matrix (calls LOC,<br>CADD) | 198                                     |
| MTRA        | transpose a matrix<br>(calls MCPY)                                        | 108                                     | RSRT        | sort matrix rows (calls<br>LOC)                          | 308                                     |
| TPRD        | transpose product<br>(calls LOC)                                          | 230                                     | CSRT        | sort matrix columns<br>(calls LOC and CCPY)              | 306                                     |
| MATA        | transpose product of<br>matrix by itself (calls<br>LOC)                   | 194                                     | RCUT        | partition row-wise<br>(calls LOC)                        | 162                                     |
| SADD        | add scalar to matrix<br>(calls LOC)                                       | 56                                      | CCUT        | partition column-wise<br>(calls LOC)                     | 162                                     |
| SSUB        | subtract scalar from a<br>matrix (calls LOC)                              | 56                                      | RTIE        | adjoin two matrices<br>row-wise (calls LOC)              | 178                                     |
| SMPY        | matrix multiplied by a<br>scalar (calls LOC)                              | 56                                      | CTIE        | adjoin two matrices<br>column-wise (calls<br>LOC)        | 166                                     |
| SDIV        | matrix divided by a<br>scalar (calls LOC)                                 | 66                                      | MCPY        | matrix copy (calls LOC)                                  | 52                                      |
| RADD        | add row of one matrix<br>to row of another<br>matrix (calls LOC)          | 90                                      | XCPY        | copy submatrix from<br>given matrix (calls LOC)          | 128                                     |
| CADD        | add column of one<br>matrix to column of<br>another matrix (calls<br>LOC) | 92                                      | RCPY        | copy row of matrix into<br>vector (calls LOC)            | 78                                      |
| SRMA        | scalar multiply row<br>and add to another row                             | 108                                     | CCPY        | copy column of matrix<br>into vector (calls LOC)         | 78                                      |
| SCMA        | scalar multiply column<br>and add to another column                       | 110                                     | DCPY        | copy diagonal of matrix<br>into vector (calls LOC)       | 58                                      |
| RINT        | interchange two rows                                                      | 94                                      | SCLA        | matrix clear and add<br>scalar (calls LOC)               | 52                                      |
| CINT        | interchange two columns                                                   | 96                                      | DCLA        | replace diagonal with<br>scalar (calls LOC)              | 50                                      |
| RSUM        | sum the rows of a matrix<br>(calls LOC)                                   | 98                                      | MSTR        | storage conversion<br>(calls LOC)                        | 116                                     |
| CSUM        | sum the columns of a<br>matrix (calls LOC)                                | 98                                      | MFUN        | matrix transformation<br>by a function                   | 66                                      |
|             |                                                                           |                                         | RECP        | reciprocal function for<br>MFUN                          | 44                                      |

| <u>Name</u>                                          | <u>Function</u>                                                                                                               | <u>Storage<br/>Required<br/>(Words)</u> | <u>Name</u>                         | <u>Function</u>                                                                                | <u>Storage<br/>Required<br/>(Words)</u> |
|------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------|-------------------------------------|------------------------------------------------------------------------------------------------|-----------------------------------------|
| LOC                                                  | location in compressed-<br>stored matrix                                                                                      | 108                                     | BESI                                | I Bessel function                                                                              | 414                                     |
| ARRAY                                                | vector storage--double<br>dimensioned conversion                                                                              | 198                                     | BESK                                | K Bessel function                                                                              | 844                                     |
| <u>Integration and Differentiation</u>               |                                                                                                                               |                                         | CEL1                                | elliptic integral of the<br>first kind                                                         | 126                                     |
| QSF                                                  | integral of tabulated<br>function                                                                                             | 806                                     | CEL2                                | elliptic integral of the<br>second kind                                                        | 200                                     |
| QATR                                                 | integral of given function<br>by trapezoidal rule.                                                                            | 386                                     | EXPI                                | exponential integral                                                                           | 262                                     |
| <u>Ordinary Differential Equations</u>               |                                                                                                                               |                                         | SICI                                | sine cosine integral                                                                           | 366                                     |
| RK1                                                  | integral of first-order<br>differential equation by<br>Runge-Kutta method (needs<br>user function subprogram)                 | 468                                     | CS                                  | Fresnel integrals                                                                              | 310                                     |
| RK2                                                  | tabulated integral of first-<br>order differential equation<br>by Runge-Kutta method<br>(needs user function sub-<br>program) | 210                                     | <u>Roots of Nonlinear Functions</u> |                                                                                                |                                         |
| RKGS                                                 | solution of a system of<br>first-order differential<br>equations by Runge-Kutta<br>method (uses given initial<br>values)      | 1174                                    | RTWI                                | refine estimate of root of<br>Wegstein's iteration (needs<br>user function subprogram)         | 208                                     |
| <u>Fourier Analysis</u>                              |                                                                                                                               |                                         | RTMI                                | determine root within a<br>range by Mueller's<br>iteration (needs user<br>function subprogram) | 536                                     |
| FORIF                                                | Fourier analysis of a<br>given function (needs user<br>function subprogram)                                                   | 292                                     | RTNI                                | refine estimate of root by<br>Newton's iteration (needs<br>user function subprogram)           | 172                                     |
| FORIT                                                | Fourier analysis of a<br>tabulated function                                                                                   | 284                                     | <u>Roots of Polynomial</u>          |                                                                                                |                                         |
| <u>Special Operations and Mathematical Functions</u> |                                                                                                                               |                                         | POLRT                               | real and complex roots<br>of polynomial                                                        | 820                                     |
| GAMMA                                                | gamma function                                                                                                                | 260                                     | <u>Polynomial Operations</u>        |                                                                                                |                                         |
| LEP                                                  | Legendre polynomial                                                                                                           | 132                                     | PADD                                | add two polynomials                                                                            | 110                                     |
| BESJ                                                 | J Bessel function                                                                                                             | 448                                     | PADDM                               | multiply polynomial by<br>constant and add to<br>another polynomial                            | 118                                     |
| BESY                                                 | Y Bessel function                                                                                                             | 704                                     | PCLA                                | replace one polynomial<br>by another                                                           | 48                                      |

| <u>Name</u> | <u>Function</u>                                                                            | <u>Storage<br/>Required<br/>(Words)</u> | <u>Name</u> | <u>Function</u>                                                         | <u>Storage<br/>Required<br/>(Words)</u> |
|-------------|--------------------------------------------------------------------------------------------|-----------------------------------------|-------------|-------------------------------------------------------------------------|-----------------------------------------|
| PSUB        | subtract one polynomial<br>from another                                                    | 112                                     | PCLD        | complete linear division                                                | 74                                      |
| PMPY        | multiply two polynomials                                                                   | 142                                     | PILD        | evaluate polynomial and<br>its first derivative (calls<br>PQSD)         | 56                                      |
| PDIV        | divide one polynomial by<br>another (calls PNORM)                                          | 198                                     | PDER        | derivative of a polynomial                                              | 88                                      |
| PQSD        | quadratic synthetic divi-<br>sion of a polynomial                                          | 78                                      | PINT        | integral of a polynomial                                                | 88                                      |
| PVAL        | value of a polynomial                                                                      | 54                                      | PGCD        | greatest common divisor<br>of two polynomials (calls<br>PDIV and PNORM) | 108                                     |
| PVSUB       | substitute variable of<br>polynomial by another<br>polynomial (calls PMPY,<br>PADDM, PCLA) | 132                                     | PNORM       | normalize coefficient<br>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 | $\chi^2$ 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                                            | AVCAL | $\Sigma$ and $\Delta$ operation                                                  |
| MFUN                                                                                                                                                                                                                                                                                                                                                                                                                         | matrix transformation by a function                           | MEANQ | mean square operation                                                            |
| RECP                                                                                                                                                                                                                                                                                                                                                                                                                         | reciprocal function for MFUN                                  | DMATX | means and dispersion matrix                                                      |
| LOC                                                                                                                                                                                                                                                                                                                                                                                                                          | location in compressed-stored matrix                          | DISCR | discriminant functions                                                           |
| CONVT                                                                                                                                                                                                                                                                                                                                                                                                                        | single precision, double precision conversion                 | LOAD  | factor loading                                                                   |
| ARRAY                                                                                                                                                                                                                                                                                                                                                                                                                        | vector storage--double dimensioned conversion                 | VARMX | varimax rotation                                                                 |
| PADD                                                                                                                                                                                                                                                                                                                                                                                                                         | add two polynomials                                           | AUTO  | autocovariances                                                                  |
| PADDM                                                                                                                                                                                                                                                                                                                                                                                                                        | multiply polynomial by constant and add to another polynomial | CROSS | crosscovariances                                                                 |
| PCLA                                                                                                                                                                                                                                                                                                                                                                                                                         | replace one polynomial by another                             | SMO   | application of filter coefficients (weights)                                     |
| PSUB                                                                                                                                                                                                                                                                                                                                                                                                                         | subtract one polynomial from another                          | EXSMO | triple exponential smoothing                                                     |
| PMPY                                                                                                                                                                                                                                                                                                                                                                                                                         | multiply two polynomials                                      | MINV  | matrix inversion                                                                 |
| PDIV                                                                                                                                                                                                                                                                                                                                                                                                                         | divide one polynomial by another                              | EIGEN | eigenvalues and eigenvectors of a real, symmetric matrix                         |
| PQSD                                                                                                                                                                                                                                                                                                                                                                                                                         | quadratic synthetic division of a polynomial                  | SIMQ  | solution of simultaneous linear, algebraic equations                             |
| PVAL                                                                                                                                                                                                                                                                                                                                                                                                                         | value of a polynomial                                         | QSF   | integral of tabulated function by Simpson's Rule                                 |
| PVSUB                                                                                                                                                                                                                                                                                                                                                                                                                        | substitute variable of polynomial by another polynomial       | QATR  | integral of given function by trapezoidal rule                                   |
| PCLD                                                                                                                                                                                                                                                                                                                                                                                                                         | complete linear division                                      | RK1   | integral of first-order differential equation by Runge-Kutta method              |
| PILD                                                                                                                                                                                                                                                                                                                                                                                                                         | evaluate polynomial and its first derivative                  | RK2   | tabulated integral of first-order differential equation by Runge-Kutta method    |
| PDER                                                                                                                                                                                                                                                                                                                                                                                                                         | derivative of a polynomial                                    | RKGS  | solution of a system of first-order differential equations by Runge-Kutta method |
| PINT                                                                                                                                                                                                                                                                                                                                                                                                                         | integral of a polynomial                                      | FORIF | Fourier analysis of a given function                                             |
| PGCD                                                                                                                                                                                                                                                                                                                                                                                                                         | greatest common divisor of two polynomials                    | FORIT | Fourier analysis of a tabulated function                                         |
| PNORM                                                                                                                                                                                                                                                                                                                                                                                                                        | normalize coefficient vector of polynomial                    | RTWI  | refine estimate of root by Wegstein's iteration                                  |
| SUBROUTINES WHOSE ACCURACY IS DATA DEPENDENT                                                                                                                                                                                                                                                                                                                                                                                 |                                                               | RTMI  | determine root within a range by Mueller's iteration                             |
| 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: |                                                               | RTNI  | refine estimate of root by Newton's iteration                                    |
|                                                                                                                                                                                                                                                                                                                                                                                                                              |                                                               | POLRT | real and complex roots of polynomial                                             |
| 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 |       |                                                                                  |

# 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  $a, a + b, a + 2b, \dots, c$  were the arguments (x) used.

| Name  | Functions              | Remarks                                                                                   | Allowable<br>Parameter<br>Range                                                                                | Range Checked<br>with<br>references*         | Maximum<br>Difference<br>s.d.=significant<br>digits<br>d.p.=decimal<br>places |
|-------|------------------------|-------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|----------------------------------------------|-------------------------------------------------------------------------------|
| GAMMA | $\Gamma(x)$ (gamma)    |                                                                                           | $x \leq 34.5$ , and $x$<br>not within $10^{-6}$<br>of zero or a<br>negative integer                            | $x = .1(1)3$                                 | 2 in 6th s.d.                                                                 |
|       |                        |                                                                                           |                                                                                                                | $x = 1(1)34$                                 | 1 in 6th s.d.                                                                 |
| LEP   | $P_n(x)$<br>(Legendre) |                                                                                           | $-1 \leq x \leq 1$<br><br>$n \geq 0$                                                                           | $x = 0(.2)1$<br>$n = 2, 3$                   | 3 in 6th s.d.                                                                 |
|       |                        |                                                                                           |                                                                                                                | $n = 9, 10$                                  | 1 in 5th s.d.                                                                 |
| BESJ  | $J_n(x)$ (Bessel)      | (The accuracy<br>Factor, D, used<br>in the program<br>was $10^{-5}$ .)                    | $x > 0; n > 0$<br><br>when $x \leq 15$ ;<br>$n < 20 + 10x^{-x^{2/3}}$<br><br>when $x > 15$ ,<br>$n < 90 + x/2$ | $x = 1(1)17$<br>$n = 0, 1, 2$                | 8 in 6th s.d.                                                                 |
|       |                        |                                                                                           |                                                                                                                | $n = 3(1)9$<br>$x = 1(1)n-2$                 | 1 in 5th s.d.                                                                 |
|       |                        |                                                                                           |                                                                                                                | $n = 3(1)9$<br>$x = n-1(1)20$                | 1 in 5th d.p.                                                                 |
|       |                        |                                                                                           |                                                                                                                | $x = 1, 2, 5, 10, 50$<br>$n = 10(10)50^{**}$ | 3 in 6th s.d.                                                                 |
| BESY  | $Y_n(x)$ (Bessel)      |                                                                                           | $n \geq 0$<br>$x > 0$                                                                                          | $x = 1(1)17$<br>$n = 0, 1, 2$                | 9 in 6th s.d.                                                                 |
|       |                        |                                                                                           |                                                                                                                | $n = 3(1)9$<br>$x = 1(1)n-2$                 | 1 in 5th s.d.                                                                 |
|       |                        |                                                                                           |                                                                                                                | $n = 3(1)9$<br>$x = n-1(1)20$                | 1 in 5th d.p.                                                                 |
|       |                        |                                                                                           |                                                                                                                | $x = 1, 2, 5, 10, 50$<br>$n = 10(10)50^{**}$ | 3 in 5th s.d.                                                                 |
| BESI  | $I_n(x)$ (Bessel)      | (Table values<br>are $e^{-x}I_n(x)$ .<br>maximum<br>difference is<br>for these<br>values) | $x > 0$<br>$0 \leq n \leq 30$                                                                                  | $x = 1(1)20$<br>$n = 0, 1$                   | 8 in 7th s.d.                                                                 |
|       |                        |                                                                                           |                                                                                                                | $x = 5(1)20$<br>$n = 2$                      | 6 in 7th s.d.                                                                 |
|       |                        |                                                                                           |                                                                                                                | $x = 1(1)20$<br>$n = 3(1)9$                  | 1 in 5th s.d.                                                                 |
|       |                        | (Table values<br>are $I_n(x)$ )                                                           |                                                                                                                | $x = 1, 2, 5, 10$<br>$n = 10, 20, 30^{**}$   | 8 in 7th s.d.                                                                 |

## Subroutines with Definite Accuracy Characteristics (continued)

| Name | Functions                                            | Remarks                                                                                 | Allowable<br>Parameter<br>Range | Range Checked<br>with<br>references*                                                                                | Maximum<br>Difference<br>s.d.=significant<br>digits<br>d.p.=decimal<br>places                              |
|------|------------------------------------------------------|-----------------------------------------------------------------------------------------|---------------------------------|---------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| BESK | $K_n(x)$ (Bessel)                                    | (Table values are $e^x K_n(x)$ . These were used for maximum differences)               | $x > 0$<br>$n \geq 0$           | $x = 1 (1) 20$<br>$n = 0, 1$<br><br>$x = 5 (1) 20$<br>$n = 2$<br><br>$x = 1 (1) 20$<br>$n = 3 (1) 9$                | 8 in 7th s.d.<br><br>9 in 7th s.d.<br><br>1 in 5th s.d.                                                    |
|      |                                                      | (Tabled values are $K_n(x)$ )                                                           |                                 | $x = 1, 2, 5, 10, 50$<br>$n = 10 (10) 50^{**}$                                                                      | 1 in 6th s.d.                                                                                              |
| CEL1 | $K(k)$<br>(elliptic 1st<br>integral)                 | (Tabled values are $K(m)$ ;<br>$m = k^2$ )                                              | $-1 \leq k \leq 1$              | $m = 0 (.1) .9$                                                                                                     | 1 in 7th s.d.                                                                                              |
| CEL2 | (Generalized<br>Integral of 2nd<br>kind)             | $K(m)$ when<br>$A = B = 1$<br>$E(m)$ when<br>$A = 1$ ,<br>$B = ck^2$<br>where $m = k^2$ | $-1 \leq k \leq 1$              | $m = 0 (.1) .9$<br><br>$m = 0 (.1) .9$                                                                              | 1 in 7th s.d.<br><br>1 in 7th s.d.                                                                         |
| EXPI | Exponential<br>Integral                              | $-Ei(-x)$<br>when $X < 0$<br><br>$E_1(x)$ when<br>$x > 0$                               | $x \geq -4$                     | $x = -.5 (.5) -2$<br><br>$x = -2.5 (-.5) -4$<br><br>$x = .5 (.5) 2$<br><br>$x = 2.5 (.5) 4$<br><br>$x = 4.5 (.5) 8$ | 0 in 7th s.d.<br><br>1 in 7th s.d.***<br><br>2 in 7th s.d.<br><br>6 in 5th s.d.***<br><br>3 in 7th s.d.*** |
| SICI | $s_i(x)$<br>(sine integral)                          |                                                                                         | none                            | $x = 1 (1) 10$<br>$x = 10 \pi$                                                                                      | 3 in 7th s.d.<br>0 in 7th s.d.                                                                             |
| SICI | $C_i(x)$<br>(cosine<br>integral)                     |                                                                                         | none                            | $x = 1 (1) 10$<br>$x = 10 \pi$                                                                                      | 3 in 7th s.d.<br>0 in 5th s.d.                                                                             |
| CS   | $C_2(u)$<br>(Fresnel)<br>$\mu = \frac{1}{2} \pi x^2$ |                                                                                         | none                            | $x = .1, .3, .6, .8$<br><br>$x = 1 (1) 5$                                                                           | 1 in 6th s.d.<br><br>2 in 7th s.d.                                                                         |
| CS   | $S_2(u)$<br>(Fresnel)<br>$\mu = \frac{1}{2} \pi x^2$ |                                                                                         | none                            | $x = .1, .3, .6, .8$<br><br>$x = 1 (1) 5$                                                                           | 1 in 4th s.d.<br><br>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 compatibility of  $x$  and  $n$ .

\*\*\*Tabled results, used for maximum difference, were given for  $xe^x E_1(-x)$  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:  $3.6 \mu s$ .)

- 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

| <u>Data Screening</u>                               | <u>Page</u> |
|-----------------------------------------------------|-------------|
| DASCR--Sample Main Program                          | 144         |
| Illustrates use of:                                 |             |
| SUBST--subset selection from observation matrix     |             |
| TAB1--tabulation of data (1 variable)               |             |
| LOC--location in compressed-stored matrix           |             |
| Special sample subroutines are:                     |             |
| BOOL--Boolean expression                            | 145         |
| HIST--histogram printing                            | 145         |
| MATIN--matrix input                                 | 145         |
| <u>Multiple Regression</u>                          |             |
| REGRE--Sample Main Program                          | 150         |
| Illustrates use of:                                 |             |
| CORRE--means, standard deviations, and correlations |             |
| ORDER--rearrangement of intercorrelations           |             |

|                                                                       |     |
|-----------------------------------------------------------------------|-----|
| MINV--matrix inversion                                                |     |
| MULTR--multiple regression                                            |     |
| Special sample subroutine is:                                         |     |
| DATA--sample data read                                                | 151 |
| <u>Polynomial Regression</u>                                          |     |
| POLRG--Sample Main Program                                            | 155 |
| Illustrates use of:                                                   |     |
| GDATA--data generation                                                |     |
| ORDER--rearrangement of intercorrelations                             |     |
| MINV--matrix inversion                                                |     |
| MULTR--multiple regression                                            |     |
| Special sample subroutine is:                                         |     |
| PLOT--output plot                                                     | 156 |
| <u>Canonical Correlation</u>                                          |     |
| MCANO--Sample Main Program                                            | 159 |
| Illustrates use of:                                                   |     |
| CORRE--means, standard deviations, and correlations                   |     |
| CANOR--canonical correlation                                          |     |
| MINV--matrix inversion                                                |     |
| NROOT--eigenvalues and eigenvectors of a special, nonsymmetric matrix |     |
| EIGEN--eigenvalues and eigenvectors of a symmetric matrix             |     |
| Special sample subroutine is:                                         |     |
| DATA--sample data read                                                | 160 |
| <u>Analysis of Variance</u>                                           |     |
| ANOVA--Sample Main Program                                            | 164 |
| Illustrates use of:                                                   |     |
| AVDAT--data storage allocation                                        |     |
| AVCAL-- $\Sigma$ and $\Delta$ operation                               |     |
| MEANQ--mean square operation                                          |     |

|                                                                 | <u>Page</u> |                                                                                                                                                                                                                                                                                                                                       | <u>Page</u> |
|-----------------------------------------------------------------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| <u>Discriminant Analysis</u>                                    |             | Illustrates use of:                                                                                                                                                                                                                                                                                                                   |             |
| MDISC--Sample Main Program                                      | 168         | QSF--numerical integration by Simpson's rule                                                                                                                                                                                                                                                                                          |             |
| Illustrates use of:                                             |             | <u>Runge-Kutta Integration</u>                                                                                                                                                                                                                                                                                                        |             |
| DMATX--means and dispersion matrix                              |             | RKINT--Sample Main Program                                                                                                                                                                                                                                                                                                            | 184         |
| MINV--matrix inversion                                          |             | Illustrates use of:                                                                                                                                                                                                                                                                                                                   |             |
| DISCR--discriminant functions                                   |             | RK2--Runge-Kutta integration                                                                                                                                                                                                                                                                                                          |             |
| <u>Factor Analysis</u>                                          |             | Special sample function is:                                                                                                                                                                                                                                                                                                           |             |
| FACTO--Sample Main Program                                      | 172         | FUN--definition of differential equation                                                                                                                                                                                                                                                                                              | 184         |
| Illustrates use of:                                             |             | <u>Real and Complex Roots of Polynomial</u>                                                                                                                                                                                                                                                                                           |             |
| CORRE--means, standard deviations, and correlations             |             | SMPRT--Sample Main Program                                                                                                                                                                                                                                                                                                            | 186         |
| EIGEN--eigenvalues and eigenvectors of a real, symmetric matrix |             | Illustrates use of:                                                                                                                                                                                                                                                                                                                   |             |
| TRACE--cumulative percentage of eigenvalues                     |             | POLRT--real and complex roots of polynomial                                                                                                                                                                                                                                                                                           |             |
| LOAD--factor loading                                            |             | <u>Solution of Simultaneous Equations</u>                                                                                                                                                                                                                                                                                             |             |
| VARMX--varimax rotation                                         |             | SOLN--Sample Main Program                                                                                                                                                                                                                                                                                                             | 190         |
| Special sample subroutine is:                                   |             | Illustrates use of:                                                                                                                                                                                                                                                                                                                   |             |
| DATA--sample data read                                          | 173         | SIMQ--solution of simultaneous equations                                                                                                                                                                                                                                                                                              |             |
| <u>Triple Exponential Smoothing</u>                             |             | LOC--location in compressed-stored matrix                                                                                                                                                                                                                                                                                             |             |
| EXPON--Sample Main Program                                      | 175         | Special sample subroutines are:                                                                                                                                                                                                                                                                                                       |             |
| Illustrates use of:                                             |             | MATIN--matrix input                                                                                                                                                                                                                                                                                                                   | 190         |
| EXSMO--triple exponential smoothing                             |             | MXOUT--matrix output                                                                                                                                                                                                                                                                                                                  | 191         |
| <u>Matrix Addition</u>                                          |             | SAMPLE PROGRAM DESCRIPTION                                                                                                                                                                                                                                                                                                            |             |
| ADSAM--Sample Main Program                                      | 179         | 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. |             |
| Illustrates use of:                                             |             | There are, however, several significant facts, which apply to all these sample programs.                                                                                                                                                                                                                                              |             |
| MADD--matrix add                                                |             | 1. Data input to programs produced by 1130                                                                                                                                                                                                                                                                                            |             |
| LOC--location in compressed-stored matrix                       |             | FORTTRAN is required to be right justified within a field, even if the data includes decimal points. <u>Only</u>                                                                                                                                                                                                                      |             |
| Special sample subroutines are:                                 |             | leading blanks are permitted.                                                                                                                                                                                                                                                                                                         |             |
| MATIN--matrix input                                             | 179         | 2. All sample programs as distributed will run                                                                                                                                                                                                                                                                                        |             |
| MXOUT--matrix output                                            | 180         | on an 8K Model IIB with 1132 Printer and 1442 Card                                                                                                                                                                                                                                                                                    |             |
| <u>Numerical Quadrature Integration</u>                         |             | Read Punch, Model 6 or 7. If the user has different                                                                                                                                                                                                                                                                                   |             |
| QDINT--Sample Main Program                                      | 182         |                                                                                                                                                                                                                                                                                                                                       |             |

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 DASCRC, the //XEQ card (which is a monitor control card), should be taken out of the routine DASCRC and placed after the \*STORE card which has stored the routine LOC on the disk. With this change, DASCRC 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, DASCRC 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, DASCRC, and six subroutines:

|       |                                              |
|-------|----------------------------------------------|
| SUBST | } are from the Scientific Subroutine Package |
| TAB1  |                                              |
| LOC   |                                              |

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:

| <u>Columns</u> | <u>Contents</u>                                     | <u>For Sample Problem</u> |
|----------------|-----------------------------------------------------|---------------------------|
| 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:

| <u>Columns</u> | <u>Contents</u>         | <u>For Sample Problem</u> |
|----------------|-------------------------|---------------------------|
| 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, DASC.R.
  - 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  $6 = 2 \times 3$ .
  - 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:

$$T = R(1) * R(2)$$

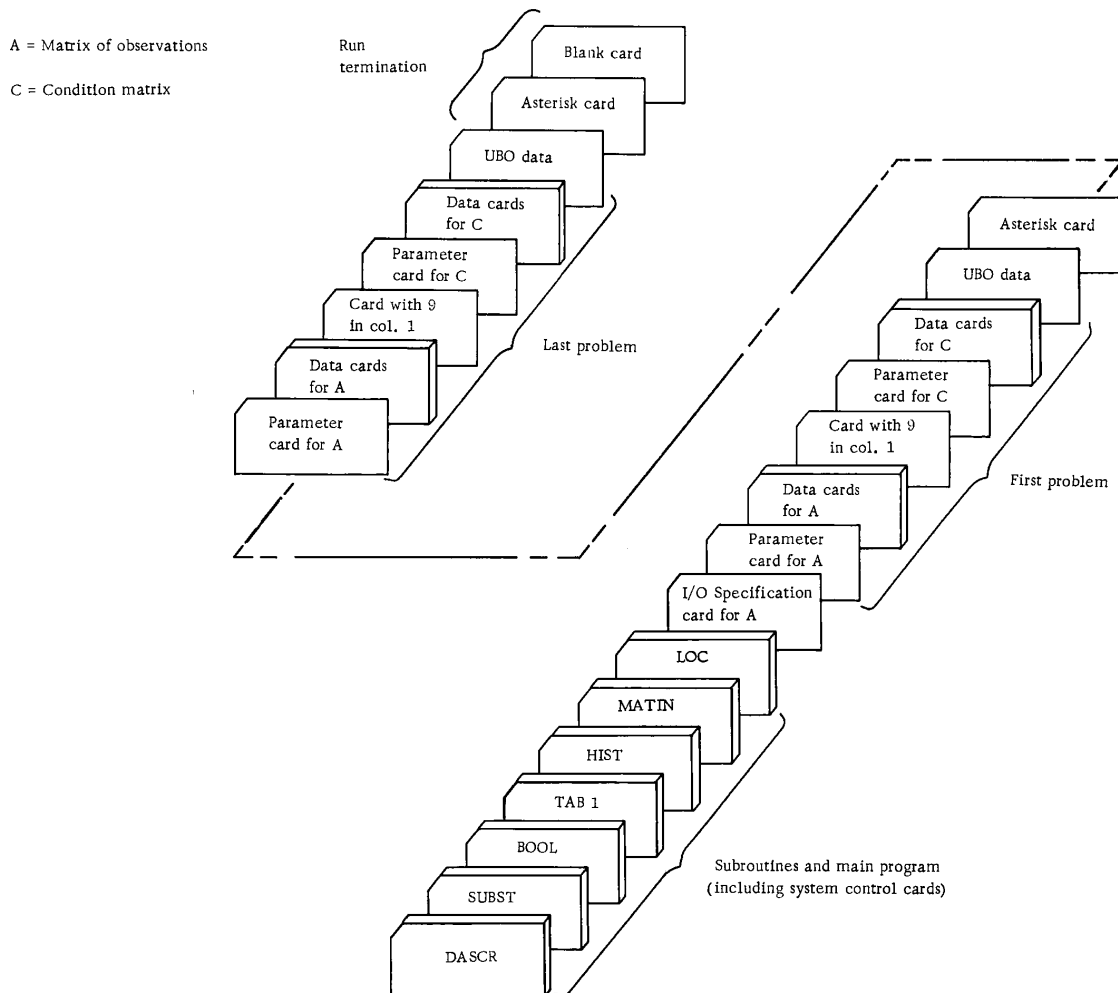


Figure 10. Deck setup (data screening)

DATA SCREENING PROBLEM 1  
SUBSET VECTOR

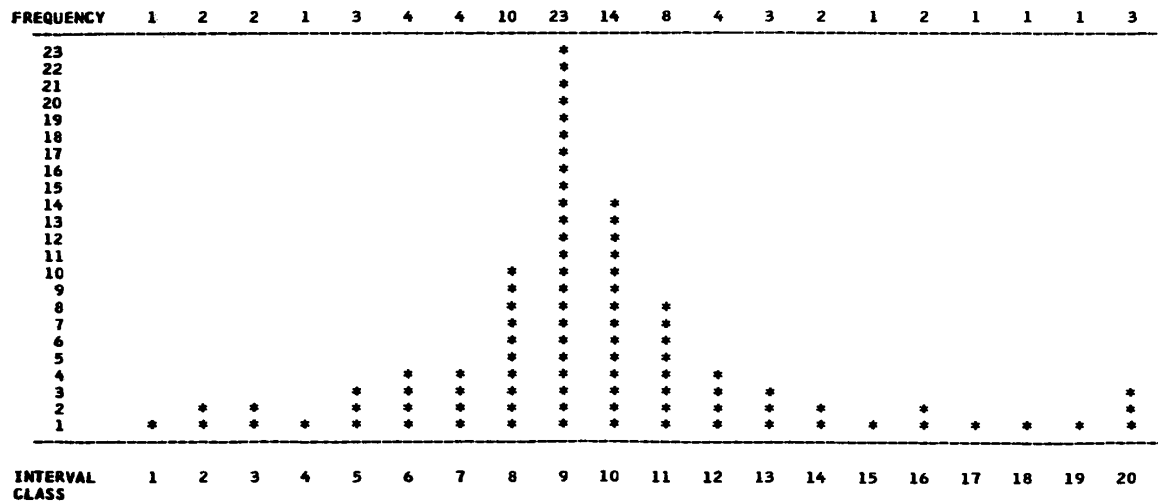
1 1.  
2 0.0  
3 1.  
4 1.  
5 1.  
6 1.  
7 1.  
8 1.  
9 1.  
10 1.  
11 1.  
12 0.0  
13 1.  
14 1.  
15 0.0  
16 1.  
17 1.  
18 1.  
19 1.  
20 1.  
21 1.  
22 1.  
23 1.  
24 1.  
25 0.0  
26 1.  
27 1.  
28 1.  
29 1.  
30 1.  
31 1.  
32 1.  
33 1.  
34 0.0  
35 1.  
36 1.  
37 1.  
38 1.  
39 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



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
//DCSCARD,TYPEWRITER,1192 PRINTER
*ONE WORD INTEGERS
C SAMPLE MAIN PROGRAM FOR DATA SCREENING - DASCR
C EXTENSAL 0001
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE
C MAXIMUM NUMBER OF ELEMENTS OF THE OBSERVATION MATRIX.
C DIMENSION A(1000)
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE
C NUMBER OF CONDITIONS TIMES 3.
C DIMENSION C(63)
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO 3.
C DIMENSION UBO(13)
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE
C NUMBER OF OBSERVATIONS.
C DIMENSION S(200)
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE
C NUMBER OF CONDITIONS.
C DIMENSION R(21)
C THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO THE
C NUMBER OF INTERVALS FOR THE SELECTED VARIABLE.
C DIMENSION FRQT(20),PCT(20)
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO 5.
C DIMENSION STATS(5)
COMMON MX,MY
10 FORMAT(///23H DATA SCREENING PROGRAM,13)
11 FORMAT(///45H DIMENSIONED AREA TOO SMALL FOR INPUT MATRIX,14)
12 FORMAT(///21H EXECUTION TERMINATED)
13 FORMAT(///43H INCORRECT NUMBER OF DATA CARDS FOR MATRIX,14)
14 FORMAT(///19H GO ON TO NEXT CASE)
15 FORMAT(///12H END OF CASE)
16 FORMAT(7F2.0,F1.0,FT.0)
17 FORMAT(3F10.0)
18 FORMAT(///14H SUBSET VECTOR,///)
19 FORMAT(///F5.0)
20 FORMAT(///23H SUMMARY STATISTICS FOR VARIABLE,13)
21 FORMAT(///6H TOTAL =F10.3,2X,9H AVERAGE =F10.3,2X,20H STANDARD DEVIATION =F10.3,2X,9H MINIMUM =F10.3,2X,9H MAXIMUM =F10.3)
22 FORMAT(12I2)
C READ I/O UNIT NUMBERS
C READ(2,22)MX,MY
24 KC=KC+1
CALL MATIN(CO,D,A,1000,NO,NV,MS,1E+1)
IF(NJ) 25,50,25
25 IF(IEC-1) 40,30,35
30 WRITE(MX,11) ICUD
WRITE(MX,14)
GO TO 24
35 WRITE(MX,13)
WRITE(MX,12)
GO TO 40
40 READ(MY,22)NC,NOVAR
JC=NC*3
READ(MY,16)IC(1),I=1,JCI
READ(MY,17)UBO(1),I=1,3)
CALL SUBST(A,C,R,BOOL,S,NO,NV,NC)
WRITE(MX,10)KC
WRITE(MX,16)
DO 50 I=1,NO
50 WRITE(MX,19)I,S(1)
CALL TAB1(A,S,NOVAR,USC,FREQ,PCT,STATS,NO,NV)
WRITE(MX,20)NOVAR
WRITE(MX,11)STATS(1),I=1,5)
JZ=UBO(2)
CALL HIST(KC,FREQ,JZ)
WRITE(MX,15)
GO TO 24
6C STOP
END
// DUP
*STORE WS UA DASCR
// KEQ UASCR
```

```
1 2
0001010000004
46 64 173 12
47 72 170 8
48 71 164 16
49 68 129 10
50 65 192 9
63 78 203 12
29 70 122 14
28 64 136 13
52 77 147 11
36 67 153 18
31 68 165 9
72 70 178 10
53 71 205 14
21 65 219 12
49 63 150 6
28 62 160 16
53 72 161 13
47 73 142 15
37 67 193 18
64 68 156 14
65 60 114 10
62 64 153 12
19 68 225 9
46 67 158 11
33 72 132 13
41 76 148 16
52 71 123 16
29 68 128 14
32 65 155 17
24 72 172 16
56 73 163 10
63 65 158 11
67 69 146 2
58 66 171 9
41 65 153 12
49 66 165 14
52 72 172 16
23 78 183 15
55 71 195 16
52 68 118 7
40 66 165 14
39 68 215 16
23 71 154 12
56 65 149 10
25 65 162 16
37 68 152 16
46 70 159 15
41 69 137 14
```

```

62 71 163 12
29 72 191 4
19 68 168 10
46 63 158 16
37 64 139 18
34 68 156 10
64 67 153 12
97 67 141 13
32 68 157 17
29 70 183 15
53 72 164 18
47 72 156 18
56 73 160 16
61 74 169 12
21 68 161 10
25 76 178 11
23 72 157 16
29 68 186 16
39 70 159 14
42 70 154 10
56 62 159 12
63 70 177 12
51 71 161 9
41 66 158 10
33 69 158 16
37 68 157 16
25 70 163 15
63 68 159 12
53 71 202 6
51 72 167 14
47 73 164 14
39 75 151 12
28 68 166 10
64 69 156 16
55 67 144 16
51 66 177 10
46 65 157 12
72 66 125 10
66 65 131 12
28 74 149 18
27 71 168 11
23 72 158 12
23 72 163 12
60 68 157 9
30 66 142 10
19 67 162 16
46 74 154 16
50 68 158 10
61 66 161 14
36 64 157 15
32 71 156 16

```

```

9 0203 65 46 8
12 120 20 210

```

#### 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(I,R)
  DIMENSION R(1)
  L1=1
  L2=2
  T=R(L1)*R(L2)
  RETURN
END

```

#### 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)
  DIMENSION JOUT(20),FREQ(20)
  COMMON MX,MY
  1 FORMAT(10H EACH ,10H EQUALS ,12,7H POINTS,/)
  2 FORMAT(10H ,4X,20(1X,A1))
  3 FPMAT(10H INTERVAL,4X,19(12,1X),12)
  4 FPMAT(10H ,4X,11H HISTOGRAM ,13)
  5 FPMAT(10H FREQUENCY,20(15))
  6 FPMAT(10H CLASS)
  7 FPMAT(11H -----)
  8 FPMAT(12H )
  9 READ(MY,1) K,NOT1
  PRINT TITLE AND FREQUENCY VECTOR
  WRITE(MX,K) NU
  DO 12 I=1,IN
  12 JOUT(I)=FREQ(I)
  WRITE(MX,51) JOUT(I),I=1,IN
  WRITE(MX,7)

```

```

C FIND LARGEST FREQUENCY
  FMAX=0.0
  DO 20 I=1,IN
  IF(FREQ(I)-FMAX) 20,20,15
  15 FMAX=FREQ(I)
  20 CONTINUE
C SCALE IF NECESSARY
  JSCAL=1
  IF(FMAX-57.0) 40,40,30
  30 JSCAL=(FMAX+49.0)/50.0
  WRITE(MX,1)K,JSCAL
C CLEAR OUTPUT AREA TO BLANKS
  40 DO 50 I=1,IN
  50 JOUT(I)=NORTH
C LOCATE FREQUENCIES IN EACH INTERVAL
  MAX=FMAX/FLOAT(JSCAL)
  DO 60 I=1,MAX
  X=MAX-(I-1)
  DO 70 J=1,IN
  IF(FREQ(J)/FLOAT(JSCAL)-X) 70,60,60
  60 JOUT(J)=K
  70 CONTINUE
  IX=X*FLOAT(JSCAL)
C PRINT LINE OF FREQUENCIES
  80 WRITE(MX,2)IX,(JOUT(J),J=1,IN)
C GENERATE CONSTANTS
  DO 90 I=1,IN
  90 JOUT(I)=1
C PRINT INTERVAL NUMBERS
  WRITE(MX,3)
  WRITE(MX,31) JOUT(I),J=1,IN
  WRITE(MX,4)
  RETURN
END

```

#### 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)
  DIMENSION A(1)
  DIMENSION CARD(8)
  COMMON MX,MY
  1 FPMAT(7F10.0)
  2 FPMAT(10,214,12)
  3 FPMAT(11)
  4 ICODE=1
  5 IER=0
  6 READ( MY,2)ICODE,IROW,ICOL,IS
  CALL LOC(IROW,ICOL,ICNT,IROW,ICOL,IS)
  IF(ISIZE-ICNT)6,7,7
  7 IF (ICNT)38,38,8
  8 ICOL=ICOL
  9 ICODE=1
  10 IER=0
  11 ICODES=ICOL-1/IC+1
  12 IF(15-115,15,12)
  13 ICODES=1
  14 SET UP LOOP FOR NUMBER OF CARDS IN ROW
  15 DO 31 K=1,ICODES
  READ(MY,11)ICARD(1),I=1,IC+1
  16 SKIP THROUGH DATA CARDS IF INPUT AREA TOO SMALL
  IF(IER)16,31
  17 L=0
  18 COMPUTE COLUMN NUMBER FOR FIRST FIELD IN CURRENT CARD
  JS=(K-1)*IC+ICOL-ICOL+1

```

```

      JE=JS+JOC-1
      IF(I5-1119,19,17
17  JF=JS
      SET UP LOOP FOR DATA ELEMENTS WITHIN CARD
19  DO 30 J=JS,JF
      IF(J-ICOL)20,20,31
20  CALL LDC(IROCR ,J,IJ,IROW,ICOL,15)
      L=L+1
30  A(I,J)=CARD(IJ)
31  CONTINUE
      IROCR=IROCR+1
      IF(IROW=IROCR) 38,35,35
35  IF(I5-1137,36,36
36  ICOLT=ICOLT-1
37  GO TO 11
38  READ(MY,3)ICARD
      IF(ICARD=9)39,40,39
39  IER=2
40  RETURN
      END

```

```

      MATIN 29
      MATIN 30
      MATIN 31
      MATIN 32
      MATIN 33
      MATIN 34
      MATIN 35
      MATIN 36
      MATIN 37
      MATIN 38
      MATIN 39
      MATIN 40
      MATIN 41
      MATIN 42
      MATIN 43
      MATIN 44
      MATIN 45
      MATIN 46
      MATIN 47
      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

| Observation | Variables      |                |                |                |                |                |
|-------------|----------------|----------------|----------------|----------------|----------------|----------------|
|             | X <sub>1</sub> | X <sub>2</sub> | X <sub>3</sub> | X <sub>4</sub> | X <sub>5</sub> | X <sub>6</sub> |
| 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 sub-routines:

|       |   |                         |
|-------|---|-------------------------|
| CORRE | } | are from the Scientific |
| ORDER |   | Subroutine Package      |
| MINV  |   |                         |
| MULTR |   |                         |

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:

- Up to 21 variables, including both independent and dependent variables.
- 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.

- (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             |

| <u>Columns</u> | <u>Contents</u>                          | <u>For Sample Problem</u> |
|----------------|------------------------------------------|---------------------------|
| 7 - 11         | Number of observations                   | 00030                     |
| 12 - 13        | Number of variables                      | 06                        |
| 14 - 15        | Number of selection cards<br>(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:

| <u>Columns</u> | <u>Contents</u>                                                                      | <u>For Sample Problem</u> | <u>Selec- tion 1</u> | <u>Selec- tion 2</u> |
|----------------|--------------------------------------------------------------------------------------|---------------------------|----------------------|----------------------|
| 1 - 2          | Option code for table of residuals<br>00 if it is not desired<br>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                   |                      |

| <u>Columns</u>  | <u>Contents</u>                                                     | <u>For Sample Problem</u> | <u>Selec- tion 1</u> | <u>Selec- tion 2</u> |
|-----------------|---------------------------------------------------------------------|---------------------------|----------------------|----------------------|
| 5 - 6<br>(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

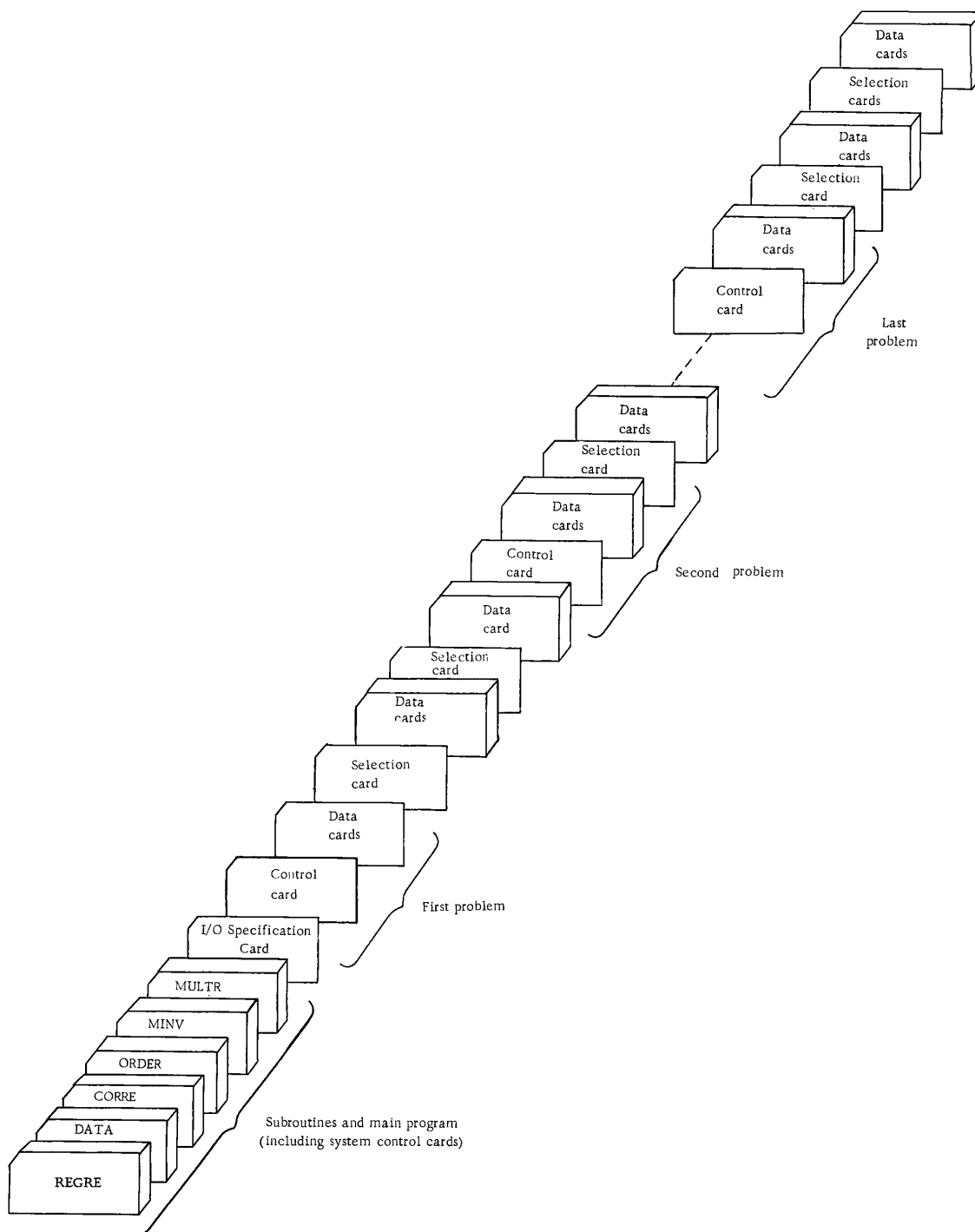


Figure 12. Deck setup (multiple linear regression)

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 MEAN STANDARD CORRELATION REGRESSION STD. ERROR COMPUTED
NO. DEVIATION A VS Y COEFFICIENT OF REG. COEF. T VALUE
1 43.13333 6.32175 0.26822 0.01242 0.03636 0.34172
2 316.16668 114.42994 0.42189 0.00738 0.00386 3.96945
3 241.80001 36.43074 0.11900 0.01504 0.00634 2.36892
4 109.66667 17.85638 0.37822 0.00150 0.03678 0.04101
5 34.13333 15.97869 0.39412 0.04918 0.04141 1.18702

DEPENDENT
6 7.26666 1.41258

INTERCEPT -5.07035
MULTIPLE CORRELATION 0.73575
STD. ERROR OF ESTIMATE 1.05161

ANALYSIS OF VARIANCE FOR THE REGRESSION
SOURCE OF VARIATION DEGREES SUM OF MEAN F VALUE
OF FREEDOM SQUARES SQUARES
ATTRIBUTABLE TO REGRESSION 5 21.32517 4.26503 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 0.51911
2 2.00000 1.77669 0.22331
3 3.00000 2.14585 0.85415
4 0.00000 0.82879 0.82879
5 2.00000 1.40322 0.59678
6 2.00000 1.92124 0.07875
7 3.00000 3.46647 0.46647
8 2.00000 2.25886 0.25886
9 3.00000 3.40238 0.40238
10 0.00000 1.02042 1.02042
11 4.00000 2.47735 1.52265
12 1.00000 2.00055 1.00055
13 1.00000 2.00735 1.00735
14 1.00000 1.15307 0.15307
15 3.00000 2.90443 0.90443
16 2.00000 1.83831 0.16668
17 3.00000 2.86004 0.43995
18 4.00000 3.43229 0.56770
19 4.00000 3.82643 0.82643
20 3.00000 2.68067 0.31932
21 4.00000 3.44886 0.44886
22 4.00000 1.80541 2.13459
23 1.00000 2.09842 1.09842
24 0.00000 1.47217 1.47217
25 4.00000 1.41253 2.58746
26 1.00000 1.88026 0.88026
27 3.00000 2.71646 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 MEAN STANDARD CORRELATION REGRESSION STD. ERROR COMPUTED
NO. DEVIATION A VS Y COEFFICIENT OF REG. COEF. T VALUE
2 316.16668 114.42994 0.42189 0.00738 0.00172 4.31764
3 241.80001 36.43074 0.11900 0.01497 0.00591 2.71643
5 34.13333 15.97869 0.39412 0.05362 0.01258 4.26263

DEPENDENT
6 7.26666 1.41258

INTERCEPT -5.93930
MULTIPLE CORRELATION 0.73423
STD. ERROR OF ESTIMATE 1.01201

ANALYSIS OF VARIANCE FOR THE REGRESSION
SOURCE OF VARIATION DEGREES SUM OF MEAN F VALUE
OF FREEDOM SQUARES SQUARES
ATTRIBUTABLE TO REGRESSION 3 21.19401 10.59807 10.13718
DEVIATION FROM REGRESSION 26 26.67265 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 1.82662 0.17337
3 2.00000 2.84619 0.26681
4 0.00000 0.80703 0.80703
5 2.00000 1.99812 0.00187
6 2.00000 1.58407 0.41592
7 3.00000 3.49850 0.49850
8 2.00000 2.23847 0.23152
9 3.00000 3.85615 0.85615
10 0.00000 0.88963 0.88963
11 4.00000 2.51254 1.48745
12 1.00000 1.95925 1.04075
13 1.00000 2.04998 1.04998
14 1.00000 1.10725 0.10725
15 3.00000 2.91951 0.08048
16 2.00000 1.74538 0.25461
17 4.00000 3.34092 0.45907
18 4.00000 3.84581 0.45418
19 4.00000 3.67661 0.32338
20 3.00000 2.85436 0.34563
21 4.00000 3.70045 0.29954
22 4.00000 1.84618 2.15381
23 1.00000 2.08899 1.08899
24 0.00000 1.59640 1.59640
25 4.00000 1.34019 2.65980
26 1.00000 1.78616 0.71383
27 3.00000 2.42442 0.75458
28 4.00000 4.12168 0.87831
29 4.00000 3.92577 0.07422
30 0.00000 0.13591 0.13591

```

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
C THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO THE
C NUMBER OF VARIABLES+1
C DIMENSION XBAR(21),STD(21),D(21),RY(21),ISAVE(21),B(21),
15B(21),T(21),W(21)
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE
C PRODUCT OF M*M
C DIMENSION RX(44)
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO
C (M+1)*M/2+
C DIMENSION R(21)
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO 10+
C DIMENSION ANS(10)
C COMMON MX,MY
C READ(2,15)MX,MY
15 FORMAT(2I2)
1 FORMATT(10,15,21)
2 FORMATT(25,M) MULTIPLE REGRESSION.....A*AZ//6X,14HSELECTION.....
112//
3 FORMATT(9H VARIABLE+5X,4HMEAN+6X,8HSTANDARD+6X,11HCORRELATION+4X,REGRE 20
110HREGRESSION+4X,10HSTD, ERROR+5X,8HCOMPUTED/6H NO+18X,9HDEVIAT+REGRE 21
210N,7X,6HX VS Y,7X,11HCOEFFICIENT+3X,12HOF REG,COEF,+3X,7HT VALUE+REGRE 22
4 FORMATT(14,6F16.5)
5 FORMATT(110H DEPENDENT)
6 FORMATT(110H INTERCEPT,13X,F13.5//23H MULTIPLE CORRELATION +F13,REGRE 25
15//23H STD, ERROR OF ESTIMATE,F13.5//)
7 FORMATT(12X,30H ANALYSIS OF VARIANCE FOR THE REGRESSION//5X,19H50REGRE 26
10HURCE OF VARIATION,7X,7HDEGREES,7X,6HSUM OF,10X,4HMEAN,13X,7HT VALUE+REGRE 28
2E/30X,10HOF FREEDOM+4X,7HSQUARES+9X,7HSQUARES)
8 FORMATT(30H ATTRIBUTABLE TO REGRESSION +16,3F16.5/30H DEVIATION REGRE 29
1FROM REGRESSION +16,2F16.5)
9 FORMATT(5X,5HTOTAL,19X,16,F16.5)
10 FORMATT(36I2)
11 FORMATT(15X,18HTABLE OF RESIDUALS//9H CASE NO+5X,7HT VALUE+5X,10HREGRE 30
1HT ESTIMATE+6X,8HRESIDUAL)
12 FORMATT(16,F15.5,2F16.5)
13 FORMATT(110H NUMBER OF SELECTIONS NOT SPECIFIED. JOB TERMINATEDREGRE 31
1)
14 FORMATT(152H THE MATRIX IS SINGULAR. THIS SELECTION IS SKIPPED+1 REGRE 38
1)
C READ PROBLEM PARAMETER CARD
100 READ (MY,1) PR,PR1,N,M,NMS
C PR.....PROBLEM NUMBER (MAY BE ALPHAMERIC)
C PR1.....PROBLEM NUMBER (CONTINUED)
C N.....NUMBER OF OBSERVATIONS
C M.....NUMBER OF VARIABLES
C NS.....NUMBER OF SELECTIONS
C
100=
C X=0
C CALL CORRE (N,M,10,X,XBAR,STD,RX,RD,B,T)
C TEST NUMBER OF SELECTIONS
C IF(NS) 108, 108, 109
108 WRITE (MX,13)
GO TO 300
109 DO 200 I=1,NS
WRITE (MX,2) PR,PR1,I
C READ SUBSET SELECTION CARD
C READ (MY,10)NRESI,NDEP,K,(ISAVE(I),J=1,K)
C NRESI.....OPTION CODE FOR TABLE OF RESIDUALS
C 0 IF IT IS NOT DESIRED,
C 1 IF IT IS DESIRED,
C NDEP.....DEPENDENT VARIABLE
C K.....NUMBER OF INDEPENDENT VARIABLES INCLUDED
C ISAVE.....A VECTOR CONTAINING THE INDEPENDENT VARIABLES
C INCLUDED
C CALL ORDER (M,N,NDEP,K,ISAVE,RX,RY)
C CALL MINV (RX,K,DET,B,T)
C TEST SINGULARITY OF THE MATRIX INVERTED
C IF(DET) 112, 110, 112
110 WRITE (MX,14)
GO TO 200
112 CALL MULTR (N,K,XBAR,STD,D,RX,RY,ISAVE,B,STD,ANS)
C PRINT MEANS, STANDARD DEVIATIONS, INTERCORRELATIONS BETWEEN
C X AND Y, REGRESSION COEFFICIENTS, STANDARD DEVIATIONS OF
C REGRESSION COEFFICIENTS, AND COMPUTED T-VALUES
C
M=M+1
WRITE (MX,3)
DO 115 J=1,K
L=ISAVE(J)
WRITE (MX,4) L,XBAR(L),STD(L),RY(J),B(J),SB(J),T(J)
WRITE (MX,5)
L=ISAVE(M)
WRITE (MX,4) L,XBAR(L),STD(L)
C PRINT INTERCEPT, MULTIPLE CORRELATION COEFFICIENT, AND
C STANDARD ERROR OF ESTIMATE
WRITE (MX,6) ANS(1),ANS(2),ANS(3)
REGRE 85
```

```

C      PRINT ANALYSIS OF VARIANCE FOR THE REGRESSION
      WRITE (MX,7)
      L=ANS(8)
      WRITE (MX,8) K=ANS(4),ANS(6),ANS(10),L=ANS(7),ANS(9)
      L=N-1
      SUM=ANS(4)+ANS(7)
      WRITE (MX,9) L,SUM
      IF (NRES) 200, 200, 120
      PRINT TABLE OF RESIDUALS
C      120 WRITE (MX,2) PR=PHI+1
      WRITE (MX,11)
      MM=ISAVE(K-1)
      DO 140 I=1,N
      CALL DATA(M,M)
      SUM=ANS(11)
      DO 130 J=1,K
      L=ISAVE(J)
      130 SUM=SUM+(L)*B(J)
      RESI=W(M)-SUM
      140 WRITE (MX,12) I,I*(MM)+SUM*RESI
      200 CONTINUE
      GO TO 100
      300 STOP
      END

```

```

HEGME 86
REGML 87
REGME 88
REGML 89
REGML 90
REGML 91
REGML 92
REGML 93
REGML 94
REGML 95
REGML 96
REGML 97
HEGME 98
REGML 99
HEGME 100
REGML 101
REGML 102
HEGME 103
REGML 104
REGML 105
HEGME 106
REGML 107
HEGME 108
REGML 109

```

**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 (N,D)
  DIMENSION D(11)
  COMMON MX,MY
  1 FORMAT(12F6.0)
  READ (MY,1) (D(I),I=1,M)
  RETURN
  END

```

```

1 2
SAMPLE0000300602
29 289 216 85 14 1
30 391 244 92 16 2
30 424 246 90 18 2
30 313 239 91 10 0
35 243 275 95 30 2
35 365 219 95 21 2
43 396 267 100 39 3
43 356 274 79 19 2
44 346 255 126 56 3
44 156 258 95 28 0
44 278 249 110 42 4
44 349 252 88 21 1
44 141 236 129 56 1
44 245 236 97 24 1
45 297 256 111 45 3
45 310 262 94 20 2
45 151 339 96 35 3
45 370 357 88 15 4
45 379 198 147 64 4
45 463 206 105 31 3
45 316 245 132 60 4
45 280 225 108 36 4
44 395 215 101 27 1
49 139 220 136 59 0
49 245 205 113 37 4
49 373 215 88 25 1
51 224 215 118 54 3
51 677 210 116 33 4
51 424 210 140 59 4
51 150 210 105 30 0
0106050102030405
29 289 216 85 14 1
30 391 244 92 16 2
30 424 246 90 18 2
30 313 239 91 10 0
35 243 275 95 30 2
35 365 219 95 21 2
43 396 267 100 39 3
43 356 274 79 19 2
44 346 255 126 56 3
44 156 258 95 28 0
44 278 249 110 42 4
44 349 252 88 21 1
44 141 236 129 56 1
44 245 236 97 24 1
45 297 256 111 45 3
45 310 262 94 20 2
45 151 339 96 35 3
45 370 357 88 15 4
45 379 198 147 64 4
45 463 206 105 31 3
45 316 245 132 60 4
45 280 225 108 36 4
44 395 215 101 27 1
49 139 220 136 59 0
49 245 205 113 37 4
49 373 215 88 25 1
51 224 215 118 54 3
51 677 210 116 33 4
51 424 210 140 59 4
51 150 210 105 30 0
010603020305
29 289 216 85 14 1
30 391 244 92 16 2
30 424 246 90 18 2
30 313 239 91 10 0
35 243 275 95 30 2
35 365 219 95 21 2
43 396 267 100 39 3
43 356 274 79 19 2
44 346 255 126 56 3
44 156 258 95 28 0
44 278 249 110 42 4
44 349 252 88 21 1
44 141 236 129 56 1
44 245 236 97 24 1
45 297 256 111 45 3
45 310 262 94 20 2
45 151 339 96 35 3
45 370 357 88 15 4
45 379 198 147 64 4
45 463 206 105 31 3
45 316 245 132 60 4
45 280 225 108 36 4
44 395 215 101 27 1
49 139 220 136 59 0
49 245 205 113 37 4
49 373 215 88 25 1
51 224 215 118 54 3
51 677 210 116 33 4
51 424 210 140 59 4
51 150 210 105 30 0

```

## 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 | } are from the Scientific Subroutine Package |
| ORDER |                                              |
| MINV  |                                              |
| MULTR |                                              |
| 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                 |

### Columns

### Contents

### For Sample Problem

14 Option code for plotting Y values and Y estimates:

1

0 if it is not desired

1 if it is desired

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.

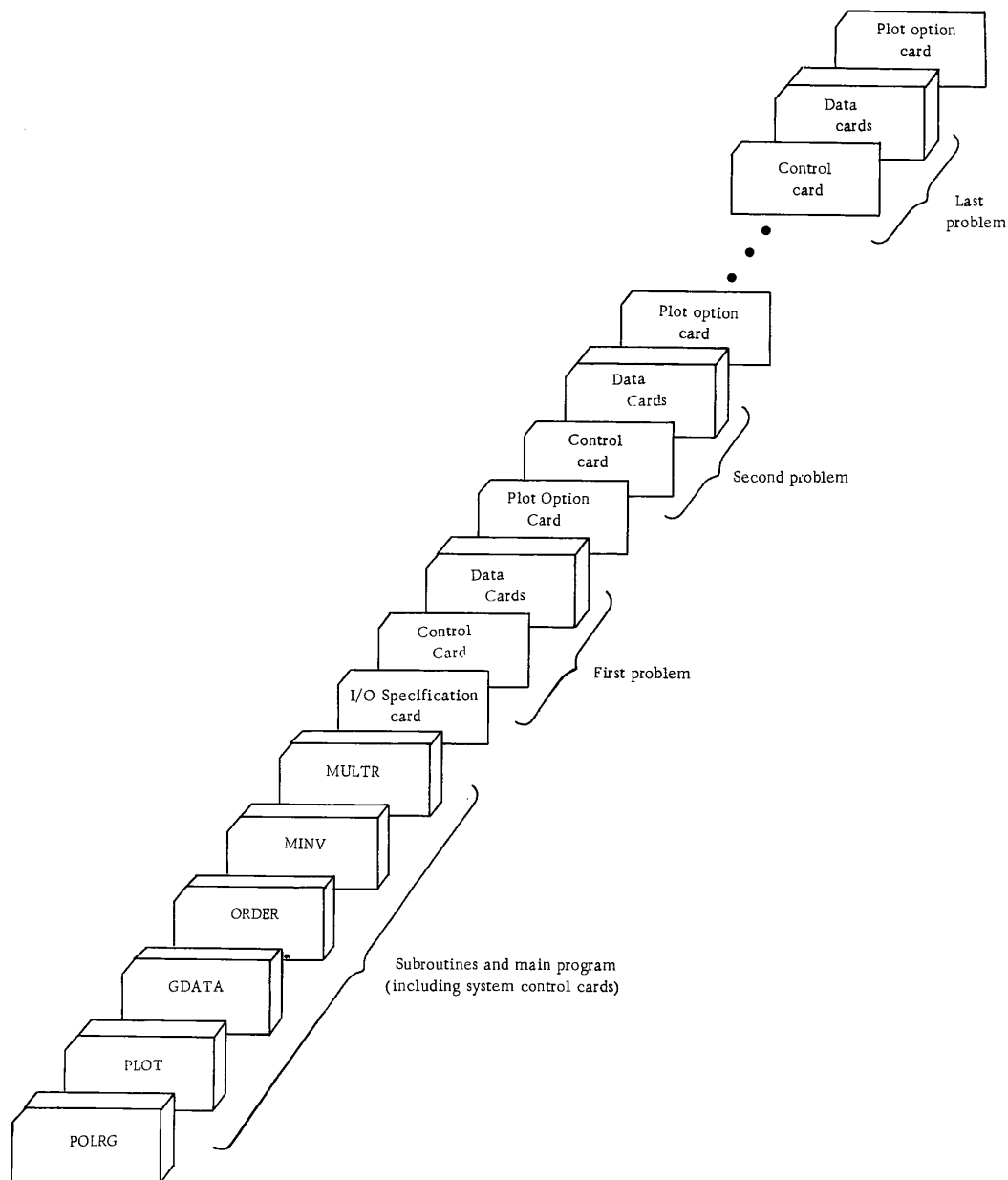


Figure 14. Deck setup (polynomial regression)

## 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  $m \times m$ . For the sample problem this product is  $16 = 4 \times 4$ .
  - 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 dimension 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).

```

POLYNOMIAL REGRESSION.....SAMPLE
NUMBER OF OBSERVATIONS: 15

POLYNOMIAL REGRESSION OF DEGREE 4
INTERCEPT: -13.87618
REGRESSION COEFFICIENTS:
 8.18781

ANALYSIS OF VARIANCE FOR 4 DEGREE POLYNOMIAL

SOURCE OF VARIATION    DEGREE OF    SUM OF    MEAN    F    IMPROVEMENT IN TERMS
                        FREEDOM      SQUARES  SQUARE  VALUE  OF SUM OF SQUARES
AGE TO REGRESSION:      1    40704.0320    40704.0320    131.68701    2.0774-20E+2
DEVIATION ABOUT REGRESSION 14    1871.49440    133.68530
TOTAL                   15
-----

POLYNOMIAL REGRESSION OF DEGREE 3
INTERCEPT: 14.07108
REGRESSION COEFFICIENTS:
-1.04071    0.04788    -0.02898

ANALYSIS OF VARIANCE FOR 3 DEGREE POLYNOMIAL

SOURCE OF VARIATION    DEGREE OF    SUM OF    MEAN    F    IMPROVEMENT IN TERMS
                        FREEDOM      SQUARES  SQUARE  VALUE  OF SUM OF SQUARES
AGE TO REGRESSION:      3    32204.38848    10734.79280    341.17708    1.882-11E+2
DEVIATION ABOUT REGRESSION 12    2291.60552    190.96712
TOTAL                   15
-----

POLYNOMIAL REGRESSION OF DEGREE 2
INTERCEPT: 14.05203
REGRESSION COEFFICIENTS:
-1.04702    0.04781    -0.02848

ANALYSIS OF VARIANCE FOR 2 DEGREE POLYNOMIAL

SOURCE OF VARIATION    DEGREE OF    SUM OF    MEAN    F    IMPROVEMENT IN TERMS
                        FREEDOM      SQUARES  SQUARE  VALUE  OF SUM OF SQUARES
AGE TO REGRESSION:      2    32204.38848    16102.19424    503.47838    1.402-11E+2
DEVIATION ABOUT REGRESSION 13    2291.60552    176.27735
TOTAL                   15
-----

POLYNOMIAL REGRESSION OF DEGREE 1
INTERCEPT: 14.05542
REGRESSION COEFFICIENTS:
15.7214    -0.18033    0.47108    -0.04487

ANALYSIS OF VARIANCE FOR 1 DEGREE POLYNOMIAL

SOURCE OF VARIATION    DEGREE OF    SUM OF    MEAN    F    IMPROVEMENT IN TERMS
                        FREEDOM      SQUARES  SQUARE  VALUE  OF SUM OF SQUARES
AGE TO REGRESSION:      1    32204.38848    32204.38848    211.37942    1.04-20E+3
DEVIATION ABOUT REGRESSION 14    2291.60552
TOTAL                   15
-----

POLYNOMIAL REGRESSION.....SAMPLE
POLYNOMIAL REGRESSION OF DEGREE 4

OBSERVATION NO.    X VALUE    Y VALUE    Y ESTIMATE    RESIDUAL
1    1.07002    8.04000    8.03319    0.00680
2    2.05000    19.04000    17.43340    1.60660
3    3.03000    21.04000    20.78493    0.25507
4    4.01000    22.04000    21.28465    0.75535
5    5.00000    23.04000    21.03552    1.00448
6    6.00000    24.04000    20.04836    0.99164
7    7.07000    24.04000    20.04836    0.00000
8    8.05000    25.04000    20.04836    0.00000
9    9.00000    26.04000    19.04836    0.00000
10   10.00000    27.04000    18.04836    0.00000
11   11.00000    28.04000    17.04836    0.00000
12   12.00000    29.04000    16.04836    0.00000
13   13.00000    30.04000    15.04836    0.00000
14   14.00000    31.04000    14.04836    0.00000
15   15.00000    32.04000    13.04836    0.00000

```

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
*JDC(CARD,TYPE,WRITE,1132,PRINT)
*ONE WORD INTEGERS
C SAMPLE MAIN PROGRAM FOR POLYNOMIAL REGRESSION = POLRG
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE
C PRODUCT OF N*(M+1), WHERE N IS THE NUMBER OF OBSERVATIONS AND
C M IS THE HIGHEST DEGREE POLYNOMIAL SPECIFIED.
C DIMENSION X(150)
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE
C PRODUCT OF M*M.
C DIMENSION D(136)
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO
C (M+2)*(M+1)/2.
C DIMENSION D(128)
C THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO M.
C DIMENSION B(1),SB(1),T(10),F(16)
C THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO
C (M+1).
C DIMENSION XBAR(7),STD(7),COE(7),SUMSQ(7),ISAVE(7)
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO 10.
C DIMENSION ANS(10)
C THE FOLLOWING DIMENSION WILL BE USED IF THE PLOT OF OBSERVED
C DATA AND ESTIMATES IS DESIRED. THE SIZE OF THE DIMENSION IN
C THIS CASE MUST BE GREATER THAN OR EQUAL TO N*3, OTHERWISE,
C THE SIZE OF DIMENSION MAY BE SET TO 1.
C DIMENSION P(150)
C COMMON MX,MY
1 FORMAT(A6,A2,I5,I2,I1)
2 FORMAT(I2F6.0)
3 FORMAT(11//27M POLYNOMIAL REGRESSION.....AN,A2/)
4 FORMAT(11//23M NUMBER OF OBSERVATION+10//)
5 FORMAT(11//32M POLYNOMIAL REGRESSION OF DEGREE+13/)
6 FORMAT(11//25M INTERCEPT+15//)
7 FORMAT(11//26M REGRESSION COEFFICIENTS+110F12.5/)
8 FORMAT(11//24M ANALYSIS OF VARIANCE FOR+14+19M DEGREE POLYNOMIAL+1APURG
31//)
9 FORMAT(11//5X+19H SOURCE OF VARIATION+7X+9H DEGREE OF+7X+6H SUM OF
19X+4H MEAN+10X+1H F+9X+20H IMPROVEMENT IN TERMS+33X+7H FREEDOM+6X+
27H SQUARES+7X+6H SQUARE+7X+5H VALUE+8X+17H OF SQUARES)
10 FORMAT(11//20H DUE TO REGRESSION+12X+16F17.5+14X+5F13.5+1F20.5)
11 FORMAT(11//20H DEVIATION ABOUT REGRESSION +16F17.5+1F14.5)
12 FORMAT(8X+5H TOTAL+19X+16F17.5//)
13 FORMAT(11//17H NO IMPROVEMENT)
14 FORMAT(11//27X+19H TAIL OF RESIDUALS//15H OBSERVATION NO.+5X+7X+
1ALUE+7X+7H VALUE+7X+10H ESTIMATE+7X+8H RESIDUAL//)
15 FORMAT(11//3X+16F18.5+1F14.5+1F17.5+1F15.5)
16 FORMAT(11//22X+
READ(10,1)MX,MY
C READ PHUBLEN PARAMETER CARD
C 100 READ (MY,1) N,M,X,N,PLOT
C PR+...PROBLEM NUMBER (MAY BE ALPHAMERIC)
C PR1+...PROBLEM NUMBER (CONTINUED)
C N+...NUMBER OF OBSERVATIONS
C M+...HIGHEST DEGREE POLYNOMIAL SPECIFIED
C NPL+...OPTION CODE FOR PLOTTING
C 0 IF PLOT IS NOT DESIRED.
C 1 IF PLOT IS DESIRED.
C PRINT PROBLEM NUMBER AND N.
WRITE (MX,3) PR,PR1
WRITE (MX,4) N
C READ INPUT DATA
L=N*M
DO 110 I=1,N
J=L+1
X(I) IS THE INDEPENDENT VARIABLE, AND X(I) IS THE DEPENDENT
VARIABLE.
110 READ (MY,2) X(I),X(I)
CALL GDATA (N,M,X,XBAR,STD,D,SUMSQ)
MM=M+1
SUM=0.0
NT=M-1
DO 200 I=1,M
ISAVE(I)=1
FORM SUBSET OF CORRELATION COEFFICIENT MATRIX
CALL ORDER (MM,D,MM+1,ISAVE,DIE)
C INVERT THE SUBMATRIX OF CORRELATION COEFFICIENTS
CALL MINV (DIE+1,DET+B,T)
CALL MULTR (N+1,XBAR,STD,SUMSQ,DIE,ISAVE,B,SB,T,ANS)
C PRINT THE RESULT OF CALCULATION
WRITE (MX,5) I
IF (ANS(7)) 140,130,110
130 SUMIP=ANS(10)+SUM
IF (SUMIP) 140, 140, 150
140 WRITE (MX,13)
GO TO 210
150 WRITE (MX,6) ANS(1)
WRITE (MX,7) (B(J),J=1,I)
WRITE (MX,8) I
WRITE (MX,9) I
SUM=ANS(4)
WRITE (MX,10) I,ANS(4),ANS(6),ANS(10),SUMIP
N1=ANS(8)
WRITE (MX,11) N1,ANS(7),ANS(9)
WRITE (MX,12) NT,SUMSQ(MM)
C SAVE COEFFICIENTS FOR CALCULATION OF Y ESTIMATES
COE(1)=ANS(1)
DO 160 J=1,I
160 COE(J+1)=B(J)
L=I
200 CONTINUE
C TEST WHETHER PLOT IS DESIRED
210 IF (NPL) 100, 100, 220
C CALCULATE ESTIMATES
220 NP3=N-N
DO 230 I=1,N
NP3=NP3+1
P(NP3)=COE(1)
L=I
DO 230 J=1,L
P(NP3+P(NP3))=X(I)*COE(J+1)
230 L=L-N
C COPY OBSERVED DATA
N2=N
L=N*M
DO 240 I=1,N
P(1)=X(I)
N2=N2+1
L=L+1
240 P(N2)=X(I)
C PRINT TABLE OF RESIDUALS
WRITE (MX,3) PR,PR1
WRITE (MX,5) LA
WRITE (MX,14)
NP2=N
NP3=N-N
DO 250 I=1,N
NP2=NP2+1
NP3=NP3+1
RESID=P(NP2)-P(NP3)
PULRG 1
PULRG 2
PULRG 3
PULRG 4
PULRG 5
PULRG 6
PULRG 7
PULRG 8
PULRG 9
PULRG 10
PULRG 11
PULRG 12
PULRG 13
PULRG 14
PULRG 15
PULRG 16
PULRG 17
PULRG 18
PULRG 19
PULRG 20
PULRG 21
PULRG 22
PULRG 23
PULRG 24
PULRG 25
PULRG 26
PULRG 27
PULRG 28
PULRG 29
PULRG 30
PULRG 31
PULRG 32
PULRG 33
PULRG 34
PULRG 35
PULRG 36
PULRG 37
PULRG 38
PULRG 39
PULRG 40
PULRG 41
PULRG 42
PULRG 43
PULRG 44
PULRG 45
PULRG 46
PULRG 47
PULRG 48
PULRG 49
PULRG 50
PULRG 51
PULRG 52
PULRG 53
PULRG 54
PULRG 55
PULRG 56
PULRG 57
PULRG 58
PULRG 59
PULRG 60
PULRG 61
PULRG 62
PULRG 63
PULRG 64
PULRG 65
PULRG 66
PULRG 67
PULRG 68
PULRG 69
PULRG 70
PULRG 71
PULRG 72
PULRG 73
PULRG 74
PULRG 75
PULRG 76
PULRG 77
PULRG 78
PULRG 79
PULRG 80
PULRG 81
PULRG 82
PULRG 83
PULRG 84
PULRG 85
PULRG 86
PULRG 87
PULRG 88
PULRG 89
PULRG 90
PULRG 91
PULRG 92
PULRG 93
PULRG 94
PULRG 95
PULRG 96
PULRG 97
PULRG 98
PULRG 99
PULRG 100
PULRG 101
PULRG 102
PULRG 103
PULRG 104
PULRG 105
PULRG 106
PULRG 107
PULRG 108
PULRG 109
PULRG 110
PULRG 111
PULRG 112
PULRG 113
PULRG 114
PULRG 115
PULRG 116
PULRG 117
PULRG 118
PULRG 119
PULRG 120
PULRG 121
PULRG 122
PULRG 123
PULRG 124
PULRG 125
```

```

250 WRITE (MX,15) 1,P(1),P(NP2),P(NP3),RESID
CALL PLOT (LA,P,N+3,0,1)
GO TO 100
END

```

```

POLRG126
POLRG127
POLRG128
POLRG129

```

```

80 I=I+1
IF (I-NL) 45,84,85
84 XPR=A(N)
GO TO 50
PRINT CROSS-VARIABLES NUMBERS
86 WRITE(MX,7)
YPR(1)=YMIN
DO 90 K=1,9
90 YPR(KN+1)=YPR(KN)+YSCAL*10.0
YPR(1)=YMAX
WRITE(MX,8)YPR(1),IP=1,11)
RETURN
END

```

```

PLOT 70
PLOT 71
PLOT 72
PLOT 73
PLOT 74
PLOT 75
PLOT 76
PLOT 77
PLOT 78
PLOT 79
PLOT 80
PLOT 81
PLOT 82

```

```

1 2
SAMPLE00015041
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
123456789

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

```

#### SUBROUTINE PLOT

##### PURPOSE

PLOT SEVERAL CROSS-VARIABLES VERSUS A BASE VARIABLE

##### USAGE

CALL PLOT (MD,A,N,M,NL,NS)

##### DESCRIPTION OF PARAMETERS

MD - COUNT 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(MD,A,N,M,NL,NS)
DIMENSION OUT(10),YPR(11),ANG(9),A(11)
COMMON MX,MY
1 FORMAT (///,A0X,7M,CHART,13,/)
2 FORMAT (1X,F11.4,5X,101A1)
3 FORMAT (2X)
5 PFORMAT(10A1)
7 FORMAT(16X,121H.
1
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69

```

```

PLOT 1
PLOT 2
PLOT 3
PLOT 4
PLOT 5
PLOT 6
PLOT 7
PLOT 8
PLOT 9
PLOT 10
PLOT 11
PLOT 12
PLOT 13
PLOT 14
PLOT 15
PLOT 16
PLOT 17
PLOT 18
PLOT 19
PLOT 20
PLOT 21
PLOT 22
PLOT 23
PLOT 24
PLOT 25
PLOT 26
PLOT 27
PLOT 28
PLOT 29
PLOT 30
PLOT 31
PLOT 32
PLOT 33
PLOT 34
PLOT 35
PLOT 36
PLOT 37
PLOT 38
PLOT 39
PLOT 40
PLOT 41
PLOT 42
PLOT 43
PLOT 44
PLOT 45
PLOT 46
PLOT 47
PLOT 48
PLOT 49
PLOT 50
PLOT 51
PLOT 52
PLOT 53
PLOT 54
PLOT 55
PLOT 56
PLOT 57
PLOT 58
PLOT 59
PLOT 60
PLOT 61
PLOT 62
PLOT 63
PLOT 64
PLOT 65
PLOT 66
PLOT 67
PLOT 68
PLOT 69

```

## 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.  $\chi^2$  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

| Observation | First set      |                |                |                | Second set     |                |                |
|-------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
|             | X <sub>1</sub> | X <sub>2</sub> | X <sub>3</sub> | X <sub>4</sub> | Y <sub>1</sub> | Y <sub>2</sub> | Y <sub>3</sub> |
| 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 | } are from the Scientific Subroutine Package |
| CANOR |                                              |
| MINV  |                                              |
| NROOT |                                              |
| EIGEN |                                              |

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             |

| Columns | Contents                                                              | For Sample Problem |
|---------|-----------------------------------------------------------------------|--------------------|
| 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.

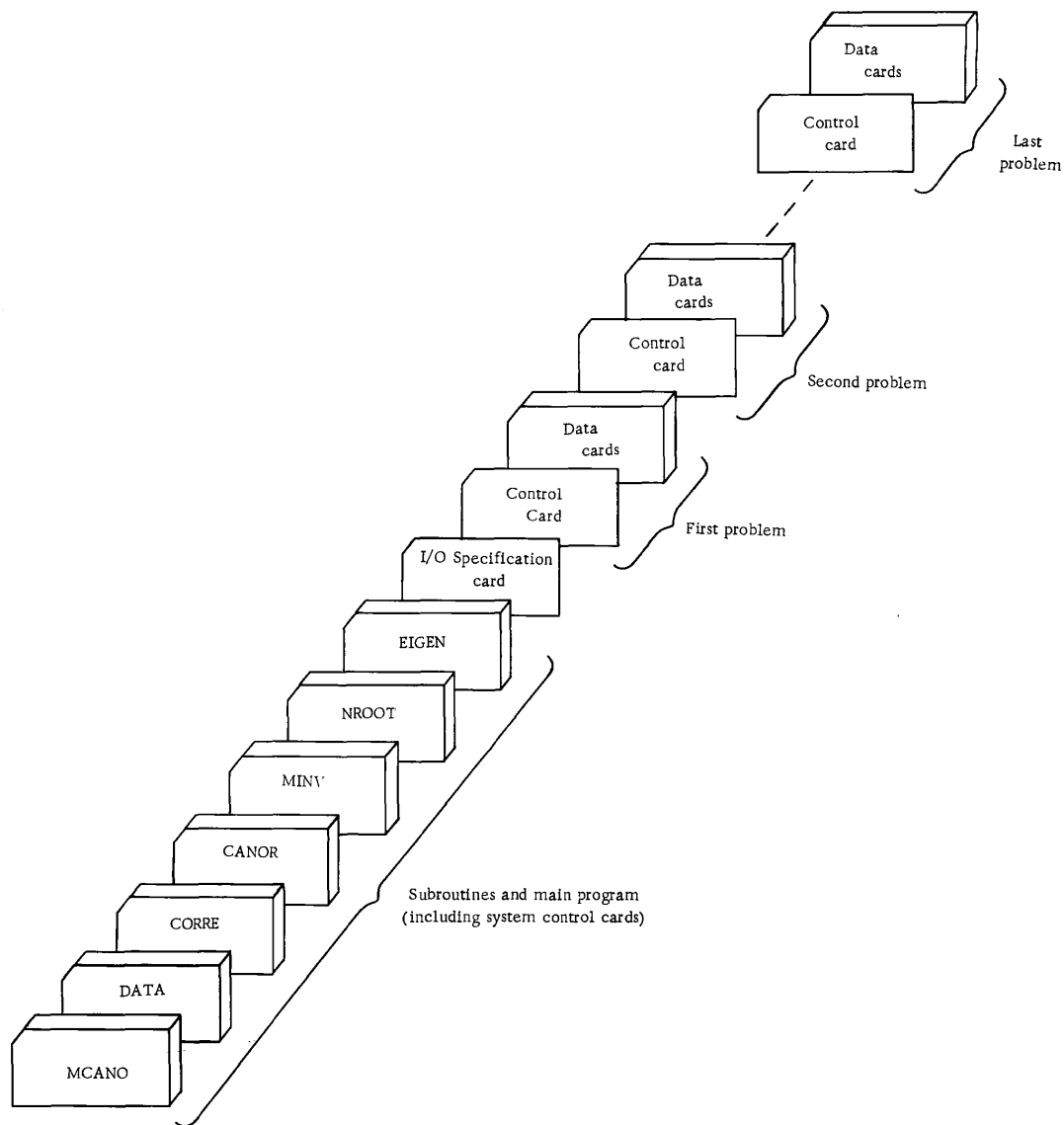


Figure 16. Deck setup (canonical correlation)

#### 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 49
NO. OF LEFT HAND VARIABLES 7
NO. OF RIGHT HAND VARIABLES 3

*****
      (1) 1.7427  (44.9135)  74.4695  37.4782  183.0500  148.8011  74.7714
STANDARD DEVIATIONS
13.1074  6.9167  10.4637  1.0816  0.8643  6.7560  9.0367

CORRELATION COEFFICIENTS
R
1.3220  0.7481  0.7582  0.6544  0.4270  0.4679  0.2482
R
0.7481  1.0000  0.4322  0.2280  0.4481  0.7179  0.3173
R
0.7582  0.4322  1.0000  0.2507  0.4794  0.4018  0.7864
R
0.6544  0.2280  0.2507  1.0000  0.3270  0.2463  0.1073
R
0.4270  0.4481  0.4794  0.3270  1.0000  0.4033  0.3927
R
0.4679  0.7179  0.4018  0.2463  0.4033  1.0000  0.4767
R
0.2482  0.3173  0.7864  0.1073  0.3927  0.4767  1.0000

NUMBER OF
EIGENVALUES
RETAINED
0
1
2

LARGEST
EIGENVALUE
0.7481
0.4322
0.2507

CORRESPONDING
CANONICAL
CORRELATION
0.8624
0.7944
0.6922

LAMBDA
0.0000
0.0000
0.0000

CHI-SQUARE
0.0000
0.0000
0.0000

DEGREES
OF
FREEDOM
12
4
2

CANONICAL CORRELATION 0.8624
COEFFICIENTS FOR LEFT HAND VARIABLES
0.8624  -0.1427  1.0321  -0.1660
COEFFICIENTS FOR RIGHT HAND VARIABLES
-0.0329  0.4481  0.4673

CANONICAL CORRELATION 0.7944
COEFFICIENTS FOR LEFT HAND VARIABLES
0.7944  -0.0329  0.4481  -0.4481
COEFFICIENTS FOR RIGHT HAND VARIABLES
-0.0329  0.4481  0.4673

CANONICAL CORRELATION 0.6922
COEFFICIENTS FOR LEFT HAND VARIABLES
0.6922  0.3207  0.1981  -0.1247
COEFFICIENTS FOR RIGHT HAND VARIABLES
0.7024  -0.1981  0.1024

```

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  $m \times m$ . For the sample problem this product is  $49 = 7 \times 7$ .
  - 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  $p \times q$ . For the sample problem this product is  $12 = 4 \times 3$ .
  - e. The dimension of array COEFR must be greater than or equal to the product of  $q \times q$ . For the sample problem this product is  $9 = 3 \times 3$ .

## 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
SAMPLE MAIN PROGRAM FOR CANONICAL CORRELATION - MCANO 1
C THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO THE MCANO 2
C TOTAL NUMBER OF VARIABLES * (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
C 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 MM. MCANO 8
C DIMENSION RX(41) MCANO 9
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO MCANO 10
C (M+1)*M*2. MCANO 11
C DIMENSION RX(45) MCANO 12
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE MCANO 13
C PRODUCT OF MP*MQ. MCANO 14
C DIMENSION COEFL(81) MCANO 15
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE MCANO 16
C PRODUCT OF MP*MQ. MCANO 17
C DIMENSION COEFL(25) MCANO 18
COMMON MX,MY MCANO 19
1 FORMAT(A4,A2,I5,2I2) MCANO 20
2 FORMAT(///27H CANONICAL CORRELATION.....A4,A2//27H NO. OF OBSERV. MCANO 21
C 3 INVENTIONS:8X,14/27H NO. OF LEFT HAND VARIABLES,15/30H NO. OF RIGHT MCANO 22
C 20HT HAND VARIABLES,14/7) MCANO 23
3 FORMAT(//20H MEANS(18F15.5)) MCANO 24
4 FORMAT(//20H STANDARD DEVIATIONS(18F15.5)) MCANO 25
5 FORMAT(//25H CORRELATION COEFFICIENTS) MCANO 26
6 FORMAT(//4H ROW,13/10F12.5)) MCANO 27
7 FORMAT(//11/12H NUMBER OF:7X,7H LARGEST,7X,13H CORRESPONDING,31X,7H CAN MCANO 28
C 10E6EES/13H EIGENVALUES,5X,10E6EENVALUE,7X,9H CANONICAL,7X,10H LARGEST MCANO 29
C 36LATION,32X,7H FREQUENCY) MCANO 30
8 FORMAT(//17/19.5F16.5,2F14.5,5X,15) MCANO 31
9 FORMAT(//22H CANONICAL CORRELATION,F12.5) MCANO 32
10 FORMAT(//30H COEFFICIENTS FOR LEFT HAND VARIABLES(18F15.5)) MCANO 33
11 FORMAT(//40H COEFFICIENTS FOR RIGHT HAND VARIABLES(18F15.5)) MCANO 34
12 FORMAT(2I2) MCANO 35
READ(2,12)MX,MY MCANO 36
C READ PROBLEM PARAMETER CARD MCANO 37
100 READ (MY,11PR,PR1,N,MP,MQ MCANO 38
C PR.....PROBLEM NUMBER (MAY BE ALPHANERIC) MCANO 39
C PR1.....PROBLEM NUMBER (CONTINUED) MCANO 40
C N.....NUMBER OF OBSERVATIONS MCANO 41
C MP.....NUMBER OF LEFT HAND VARIABLES MCANO 42
C MQ.....NUMBER OF RIGHT HAND VARIABLES MCANO 43
C WRITE (MX,21PR,PR1,N,MP,MQ MCANO 44
C 10=0 MCANO 45
C 2=0 MCANO 46
C CALL CORRE (N,M,10,XBAR,STD,RX,CANR,CHISQ,COEFL) MCANO 47
C PRINT MEANS, STANDARD DEVIATIONS, AND CORRELATION MCANO 48
C COEFFICIENTS OF ALL VARIABLES MCANO 49
C WRITE (MX,31)XBAR(1),1=1,M) MCANO 50
C WRITE (MX,41)STD(1),1=1,M) MCANO 51
C WRITE (MX,51) MCANO 52
C DO 160 I=1,M MCANO 53
C DO 150 J=1,M MCANO 54
C IF(I-J) 120, 130, 130 MCANO 55
C GO TO 140 MCANO 56
C 120 L=J+1-J/2 MCANO 57
C GO TO 140 MCANO 58
C 130 L=J+1-J/2 MCANO 59
C 140 CANR(J)=R(L) MCANO 60
C 150 CONTINUE MCANO 61
C 160 WRITE (MX,61),(CANR(J),J=1,M) MCANO 62
C CALL CANOR (N,MP,MQ,R,XBAR,STD,CANR,CHISQ,NDF,COEFL,RX) MCANO 63
C PRINT EIGENVALUES, CANONICAL CORRELATIONS, LAMBDA, CHI-SQUARES MCANO 64
C DEGREES OF FREEDOMS MCANO 65
C WRITE (MX,71) MCANO 66
C DO 170 I=1,MQ MCANO 67
C N1=N-1 MCANO 68
C TEST WHETHER EIGENVALUE IS GREATER THAN ZERO MCANO 69
C IF(XBAR(1)) 165, 165, 170 MCANO 70
C 165 MM=N1 MCANO 71
C GO TO 175 MCANO 72
C 170 WRITE (MX,81N1,XBAR(1),CANR(1),STD(1),CHISQ(1),NDF(1) MCANO 73
C MM=MQ MCANO 74
C PRINT CANONICAL COEFFICIENTS MCANO 75
C 175 N1=0 MCANO 76
C N2=0 MCANO 77
C DO 200 I=1,MM MCANO 78
C WRITE (MX,91)CANR(1) MCANO 79
C DO 180 J=1,MP MCANO 80
C N1=N1+1 MCANO 81
C 180 XBAR(J)=COEFL(N1) MCANO 82
C WRITE (MX,101)(XBAR(J),J=1,MP) MCANO 83
C DO 190 J=1,MQ MCANO 84
C N2=N2+1 MCANO 85
C 190 XBAR(J)=COEFL(N2) MCANO 86
C WRITE (MX,111)(XBAR(J),J=1,MQ) MCANO 87
C 200 CONTINUE MCANO 88
C GO TO 100 MCANO 89
C END MCANO 90
// DUP
*STORE WS UA MCANO
// XEQ MCANO 01
*LOCAL MCANO,CORRE,CANOR
1 2
SAMPLE000230403
191 155 65 19 179 145 70
195 149 70 20 201 152 69
183 148 71 19 185 149 75
183 153 82 18 188 149 86
176 144 67 18 171 142 71
208 157 81 22 192 152 77
189 150 75 21 190 149 72
197 159 90 20 189 152 82
188 152 76 19 197 159 84
192 150 78 20 187 151 72
179 158 99 18 186 148 89
183 147 65 18 174 147 70
174 150 71 19 182 152 65
190 159 91 19 195 157 99
188 151 98 20 187 158 87
```

```
163 137 59 18 161 130 63
195 155 85 20 183 158 81
196 153 80 21 173 148 74
181 145 77 20 182 146 70
175 140 70 19 145 137 61
192 154 69 20 185 152 63
174 143 79 20 178 147 73
176 139 70 20 176 143 69
```

## 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,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 <sub>1</sub> |                |                |                | b <sub>2</sub> |                |                |                | b <sub>3</sub> |                |                |                |
|---------------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
|                     |                | a <sub>1</sub> | a <sub>2</sub> | a <sub>3</sub> | a <sub>4</sub> | a <sub>1</sub> | a <sub>2</sub> | a <sub>3</sub> | a <sub>4</sub> | a <sub>1</sub> | a <sub>2</sub> | a <sub>3</sub> | a <sub>4</sub> |
| r <sub>1</sub> .... | c <sub>1</sub> | 3              | 10             | 9              | 8              | 24             | 8              | 9              | 3              | 2              | 8              | 9              | 8              |
|                     | c <sub>2</sub> | 4              | 12             | 3              | 9              | 22             | 7              | 16             | 2              | 2              | 2              | 7              | 2              |
|                     | c <sub>3</sub> | 5              | 10             | 5              | 8              | 23             | 9              | 17             | 3              | 2              | 8              | 6              | 3              |
| r <sub>2</sub> .... | c <sub>1</sub> | 2              | 14             | 9              | 13             | 29             | 16             | 11             | 3              | 2              | 7              | 5              | 3              |
|                     | c <sub>2</sub> | 7              | 11             | 5              | 8              | 28             | 18             | 10             | 6              | 6              | 6              | 5              | 9              |
|                     | c <sub>3</sub> | 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:

$$x_{abcr} \quad \text{where} \quad \begin{aligned} a &= 1, 2, 3, 4 \\ b &= 1, 2, 3 \\ c &= 1, 2, 3 \\ r &= 1, 2 \end{aligned}$$

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 | } are from the Scientific Subroutine Package |
| AVCAL |                                              |
| MEANQ |                                              |

##### 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:

$$T = \prod_{i=1}^k (\text{LEVEL}_i + 1)$$

where  $\text{LEVEL}_i$  = number of levels of  $i^{\text{th}}$  factor  
 $k$  = number of factors  
 $\Pi$  = 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.

| <u>Columns</u> | <u>Contents</u>                         |
|----------------|-----------------------------------------|
| { 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.

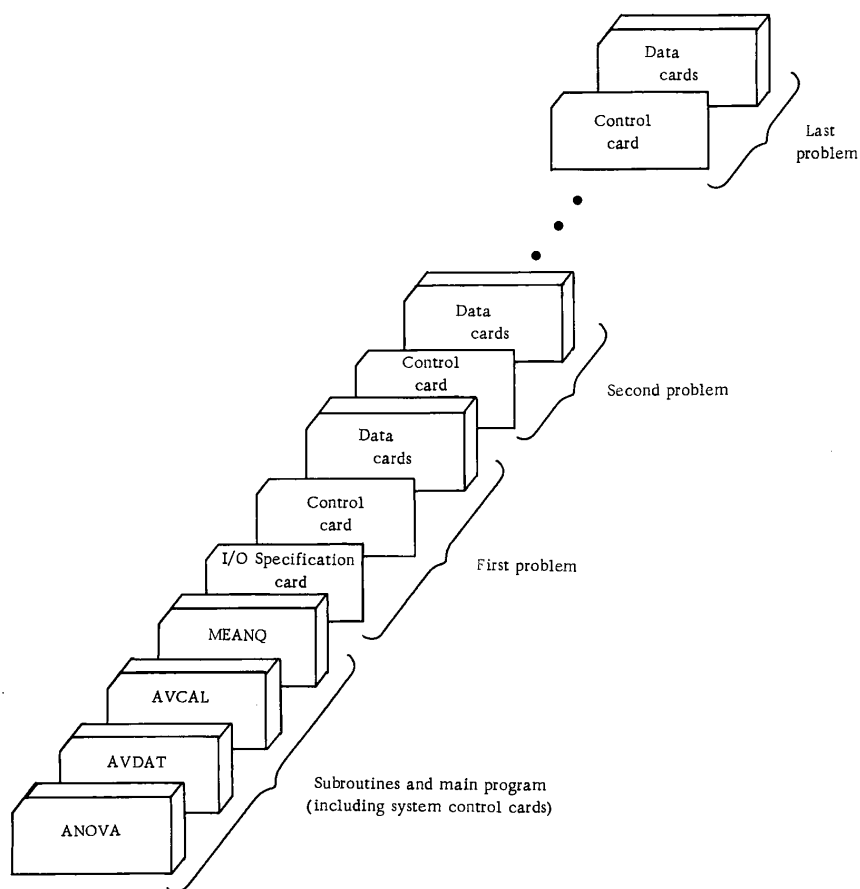


Figure 18. Deck setup (analysis of variance)

## 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<br>with Replicates | Two-way Classification<br>with Cell Replicates | Randomized<br>Complete Block<br>with Two Factors                | Split Plot                                                                                    |
|-----------------------------------|------------------------------------------|------------------------------------------------|-----------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| (Input)<br>Factor No. 1<br>2<br>3 | Groups = A<br>Replicates = R             | Rows = A<br>Columns = B<br>Replicates = R      | Factor 1 = A<br>Factor 2 = B<br>Blocks = R                      | Main treatment = A<br>Subtreatment = B<br>Blocks = R                                          |
| (Output)<br>Sums of squares       | A<br>R<br>AR                             | A<br>B<br>AB<br>R<br>AR<br>BR<br>ABR           | A<br>B<br>AB<br>R<br>AR<br>BR<br>ABR                            | A<br>B<br>AB<br>R<br>AR<br>BR<br>ABR                                                          |
| Summary<br>instruction            | Error = R + (AR)                         | Error = R + (AR)<br>+ (BR) + (ABR)             | Error = (AR) + (BR)<br>+ (ABR)                                  | Error = (BR) + (ABR)<br>(b)                                                                   |
| Analysis of<br>variance           | Groups A<br>Error                        | Rows A<br>Columns B<br>Interaction AB<br>Error | Factor 1 A<br>Factor 2 B<br>Interaction AB<br>Blocks R<br>Error | Main treatment A<br>Blocks R<br>Error (a) AR<br>Subtreatment B<br>Interaction AB<br>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     |
| R                   | 722.69945       | 2                  | 361.34722    |
| AB                  | 1382.08349      | 6                  | 230.34722    |
| C                   | 55.11111        | 2                  | 27.55555     |
| RC                  | 42.00000        | 6                  | 7.00000      |
| 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        | 6                  | 10.43982     |
| ABCR                | 151.02760       | 12                 | 12.58554     |
| 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:

- Changes in the dimension statements of the main program, ANOVA:
  - 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  $240 = (4+1)(3+1)(3+1)(2+1)$ .
  - 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.
  - 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  $15 = 2^4 - 1$ .
- 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
C DIMENSION X(1600) ANOVA502
C THE FOLLOWING DIMENSIONS MUST BE GREATER THAN OR EQUAL TO THE ANOVA 6
C NUMBER OF FACTORS.. ANOVA 7
C 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
C DIMENSION SUMSQ(63),NDF(63),SMEAN(63) ANOVA101
C THE FOLLOWING DIMENSION IS USED TO PRINT FACTOR LABELS IN ANOVA 12
C ANALYSIS OF VARIANCE TABLE AND IS FIXED ANOVA 13
C DIMENSION FMT(15) ANOVA 14
1 FORMAT(A4,A2,I2,A4,3X,11(A1,I4)/(A1,I4,A1,I4,A1,I4,A1,I4)) ANOVA 15
2 FORMAT(////20H ANALYSIS OF VARIANCE....A4,A2//) ANOVA 16
3 FORMAT(////10H LEVELS OF FACTORS/(3X,A1,7X,I4)) ANOVA 17
4 FORMAT(////11H GRAND MEAN,F20.5//) ANOVA 18
5 FORMAT(////10H SOURCE OF,18X,THSUMS OF,10X,10HDEGREES OF,9X,4HMEAN/ ANOVA 19
10H VARIATION,18X,THSQUARES,11X,THFREEDOM,10X,THSQUARES//) ANOVA 20
6 FORMAT(2X,15A1,F20.5,10X,16,F20.5) ANOVA 21
7 FORMAT(//7H TOTAL,10X,F20.5,10X,16) ANOVA 22
8 FORMAT(12F6.0) ANOVA 23
9 FORMAT(12I2) ANOVA 24
C ..... ANOVA 25
```

```
READ(2,9)MX,MY ANOVA 26
C READ PROBLEM PARAMETER CARD ANOVA 27
100 READ (MY,1)PR,PRI,K,BLANK,(HEAD(I),LEVEL(I),I=1,K) ANOVA 28
C PR,...PROBLEM NUMBER (MAY BE ALPHAMERIC) ANOVA 29
C PRI,...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
C WRITE (MX,2)PR,PRI ANOVA 36
C 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
C READ (MT,3)(AT(I),I=1,N) ANOVA 43
C CALL AVDAT (K,LEVEL,N,X,L,ISTEP,KOUNT) ANOVA 44
C CALL AVCAL (K,LEVEL,X,L,ISTEP,LASTS) ANOVA 45
C CALL MEANQ (K,LEVEL,X,GMEAN,SUMSQ,NDF,SMEAN,ISTEP,KOUNT,LASTS) ANOVA 46
C PRINT GRAND MEAN ANOVA 47
C WRITE (MX,4)GMEAN ANOVA 48
C PRINT ANALYSIS OF VARIANCE TABLE ANOVA 49
C 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)=I ANOVA 54
DO 110 I=1,15 ANOVA 55
110 FMT(I)=BLANK ANOVA 56
NN=0 ANOVA 57
SUM=0 ANOVA 58
N=N*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(I*NN) ANOVA 68
IF(NN-LL) 145, 170, 170 ANOVA 69
145 DO 150 I=L,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 140 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 ANOVA
*STORE 45 JA ANOVA
// REQ ANOVA
```

| 1   | 2  | 3        | 4 | 5                    | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |    |
|-----|----|----------|---|----------------------|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|----|
| 1 2 |    | SAMPLE04 |   | A0064B0003C0003R0002 |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |    |
| 3   | 10 | 9        | 8 | 24                   | 8 |   |   | 9 | 3  | 2  | 8  | 9  | 8  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | </ |

Table 7. Sample Data for Discriminant Analysis

|         | Observation | X <sub>1</sub> | X <sub>2</sub> | X <sub>3</sub> | X <sub>4</sub> | X <sub>5</sub> | X <sub>6</sub> |
|---------|-------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 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

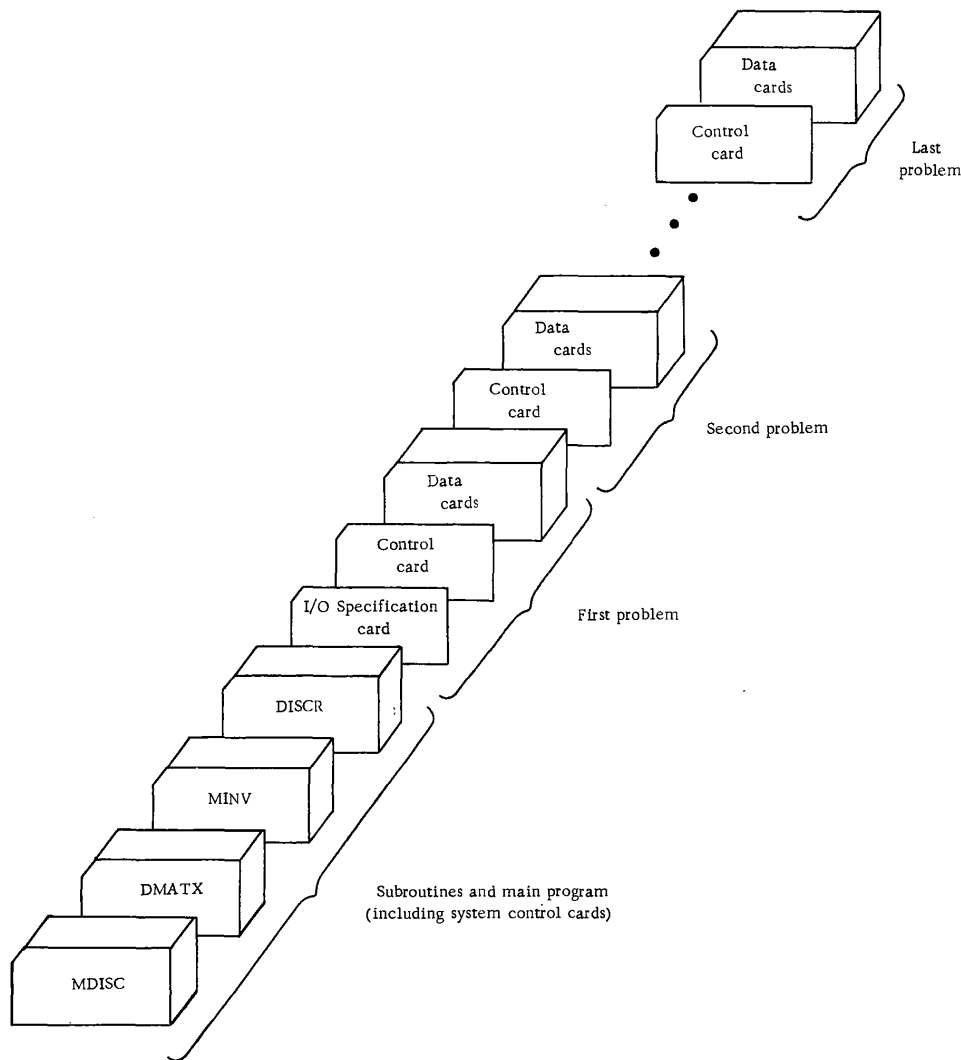


Figure 20. Deck setup (discriminant analysis)

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  $24 = 6 \times 4$ .
  - 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  $36 = 6 \times 6$ .
  - 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 SIZE=
GROUP
1
2
3
4
8
7
7
8

GROUP 1 MEANS
7.87500 7.90000 4.62500 7.25000 18.50000 9.87500

GROUP 2 MEANS
7.14285 8.57143 9.57143 7.85714 20.14286 12.57143

GROUP 3 MEANS
7.65714 7.85714 8.85714 9.28571 17.42857 10.14285

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 18.13487 9.34566
ROW 4 -6.09889 1.91758 3.93681 9.83309 4.62156 3.83790
ROW 5
ROW 6
-22.74861 22.60987 18.13487 4.62156 62.78635 30.18268
ROW 7 -9.54051 10.66757 9.34566 3.83790 30.18268 29.57484
COMMON MEANS
7.88889 7.96666 7.33333 7.89999 19.59999 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 -0.46476

DISCRIMINANT FUNCTION 2
CONSTANT * COEFFICIENTS
-29.21008 * 2.61928 2.25227 -0.04816 1.80318 0.43732 -0.21783

DISCRIMINANT FUNCTION 3
CONSTANT * COEFFICIENTS
-31.86424 * 2.74448 2.39585 -0.06456 2.13259 0.42619 -0.32718

DISCRIMINANT FUNCTION 4
CONSTANT * COEFFICIENTS
-30.82023 * 2.71858 2.03924 -0.13351 1.94538 0.71677 -0.48760

EVALUATION OF CLASSIFICATION FUNCTIONS FOR EACH OBSERVATION
GROUP 1 PROBABILITY ASSOCIATED WITH LARGEST DISCRIMINANT FUNCTION NO.
OBSERVATION 1 0.38065 4
2 0.37043 1
3 0.36280 1

GROUP 2 PROBABILITY ASSOCIATED WITH LARGEST DISCRIMINANT FUNCTION NO.
OBSERVATION 4 0.44189 1
5 0.34453 1
6 0.44216 3
7 0.37186 2
8 0.29272 2

GROUP 3 PROBABILITY ASSOCIATED WITH LARGEST DISCRIMINANT FUNCTION NO.
OBSERVATION 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 4 PROBABILITY ASSOCIATED WITH LARGEST DISCRIMINANT FUNCTION NO.
OBSERVATION 1 0.67612 3
2 0.46828 2
3 0.56834 2
4 0.66889 3
5 0.30599 2
6 0.33063 4
7 0.39006 3

GROUP 5 PROBABILITY ASSOCIATED WITH LARGEST DISCRIMINANT FUNCTION NO.
OBSERVATION 1 0.37127 4
2 0.37474 1
3 0.62340 4
4 0.50896 1
5 0.52179 4
6 0.34061 4
7 0.43135 4
8 0.27848 1
```

Figure 21. Output listing

times m. For the sample this product is  $180 = 30 \times 6$ .

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
C 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
C 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
C 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
C 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
C 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
C DIMENSION X(1000) MDISC 23
C ..... MDISC 24
1 FORMAT(A4,Z2,I2,12I5/(14I5)) MDISC 25
2 FORMAT(///27H DISCRIMINANT ANALYSIS.....A4,A2//19H NUMBER OF GROUPS MDISC 26
10UPS:TX,13/22H NUMBER OF VARIABLES,17/17H SAMPLE SIZES.../12X, MDISC 27
23HGROUP) MDISC 28
3 FORMAT(12X,13,8X,14) MDISC 29
4 FORMAT(//2X) MDISC 30
5 FORMAT(12F6.0) MDISC 31
6 FORMAT(//6H GROUP,13,7H MEANS/(8F15.5)) MDISC 32
7 FORMAT(//25H POOLED DISPERSION MATRIX) MDISC 33
8 FORMAT(//6H COM,13/(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,13/6X,27HCONSTANT * COEFF MDISC 37
11GENT5/F15.5,7H * /F15.5/22X,7F15.5) MDISC 38
12 FORMAT(//66H EVALUATION OF CLASSIFICATION FUNCTIONS FOR EACH OBS MDISC 39
12ERVATION) MDISC 40
13 FORMAT(//6H GROUP,13/19X,27HPROBABILITY ASSOCIATED WITH,11X,7HLAGRANGE MDISC 41
13TEST/13H OBSERVATION,5X,29HLARGEST DISCRIMINANT FUNCTION,8X,12HFUNN MDISC 42
2CTION NO.) MDISC 43
14 FORMAT(17/20X,F8.5,20X,16) MDISC 44
15 FORMAT(12I) MDISC 45
C ..... MDISC 46
READ(12,15)MX,MY MDISC 47
C READ PROBLEM PARAMETER CARD MDISC 48
100 READ(MY,1)PR,PR1,K,M,N(1),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
C WRITE (MX,2) PR,PR1,K,M MDISC 55
C DO 110 I=1,K MDISC 56
110 WRITE (MX,3) I,N(1) MDISC 57
C WRITE (MX,4) MDISC 58
C READ DATA MDISC 59
C DO 130 I=1,K MDISC 60
130 DO 140 J=1,M MDISC 61
N1=N(1) MDISC 62
DO 120 J=1,N1 MDISC 63
READ (MY,5) (CMEAN(IJ),J=1,M) MDISC 64
L=L+1 MDISC 65
N2=N(1) MDISC 66
DO 120 J=1,M MDISC 67
N2=N2+N MDISC 68
120 X(N2)=CMEAN(IJ) MDISC 69
130 L=N2 MDISC 70
C CALL DMATX (K,M,N,X,XBAR,D,CMEAN) MDISC 71
C PRINT MEANS AND POOLED DISPERSION MATRIX MDISC 72
C 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(I) MDISC 77
150 WRITE (MX,6) I,(CMEAN(J),J=1,M) MDISC 78
C WRITE (MX,7) MDISC 79
DO 170 I=1,M MDISC 80
L=L+M MDISC 81
DO 160 J=1,M MDISC 82
L=L+M MDISC 83
160 CMEAN(J)=DEL MDISC 84
170 WRITE (MX,8) I,(CMEAN(J),J=1,M) MDISC 85
C CALL MINV (D,M,DET,CMEAN,C) MDISC 86
C CALL DISCR (K,M,N,X,XBAR,D,CMEAN,V,C,P,LG) MDISC 87
C PRINT COMMON MEANS MDISC 88
C WRITE(MX,9) (CMEAN(I),I=1,M) MDISC 89
C PRINT GENERALIZED MAHALANOBIS D-SQUARE MDISC 90
C WRITE (MX,10) V MDISC 91
C PRINT CONSTANTS AND COEFFICIENTS OF DISCRIMINANT FUNCTIONS MDISC 92
C N1=1 MDISC 93
N2=N(1) MDISC 94
DO 180 I=1,K MDISC 95
WRITE (MX,11) I,(C(I),J=N1,N2) MDISC 96
N1=N1+N(1) MDISC 97
180 N2=N2+N(1) MDISC 98
C PRINT EVALUATION OF CLASSIFICATION FUNCTIONS FO EACH MDISC 99
C OBSERVATION MDISC 100
C WRITE (MX,12) MDISC 101
N1=1 MDISC 102
N2=N(1) MDISC 103
DO 210 I=1,K MDISC 104
WRITE (MX,13) I MDISC 105
L=0 MDISC 106
DO 190 J=N1,N2 MDISC 107
L=L+1 MDISC 108
190 WRITE (MX,14) L,P(J),LG(J) MDISC 109
C IF(I=K) 20C, 1CC, 1CO MDISC 110
20C N1=N1+N(1) MDISC 111
N2=N2+N(1) MDISC 112
210 CONTINUE MDISC 113
STOP MDISC 114
END MDISC 115
// DUP MDISC 116
*STORE WS UA MDISC 117
// REQ MDISC 01 MDISC 118
*LOCALMDISC,DMATX,MINV,DISCR MDISC 119

```

```

1 2
SAMPLE040600008000070000700008 1
3 10 9 8 24 2
4 12 3 8 22 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 8
2 10 9 8 29 16 9
7 10 9 8 28 10 10
9 10 27 8 28 16 11
11 7 8 9 8 15 12
8 10 2 8 27 16 13
1 6 8 14 14 14 14
7 8 9 6 18 2 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 | 9  | 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 <sub>1</sub> | X <sub>2</sub> | X <sub>3</sub> | X <sub>4</sub> | X <sub>5</sub> | X <sub>6</sub> | X <sub>7</sub> | X <sub>8</sub> | X <sub>9</sub> |
|-------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 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  
TRACE  
LOAD  
VARMX  
DATA

are from the Scientific Subroutine Package  
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             |

| <u>Columns</u>    | <u>Contents</u>                                                                                                                                            | <u>For Sample Problem</u> |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|
| 14 - 19<br>(cont) | correlation coefficients.<br>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.

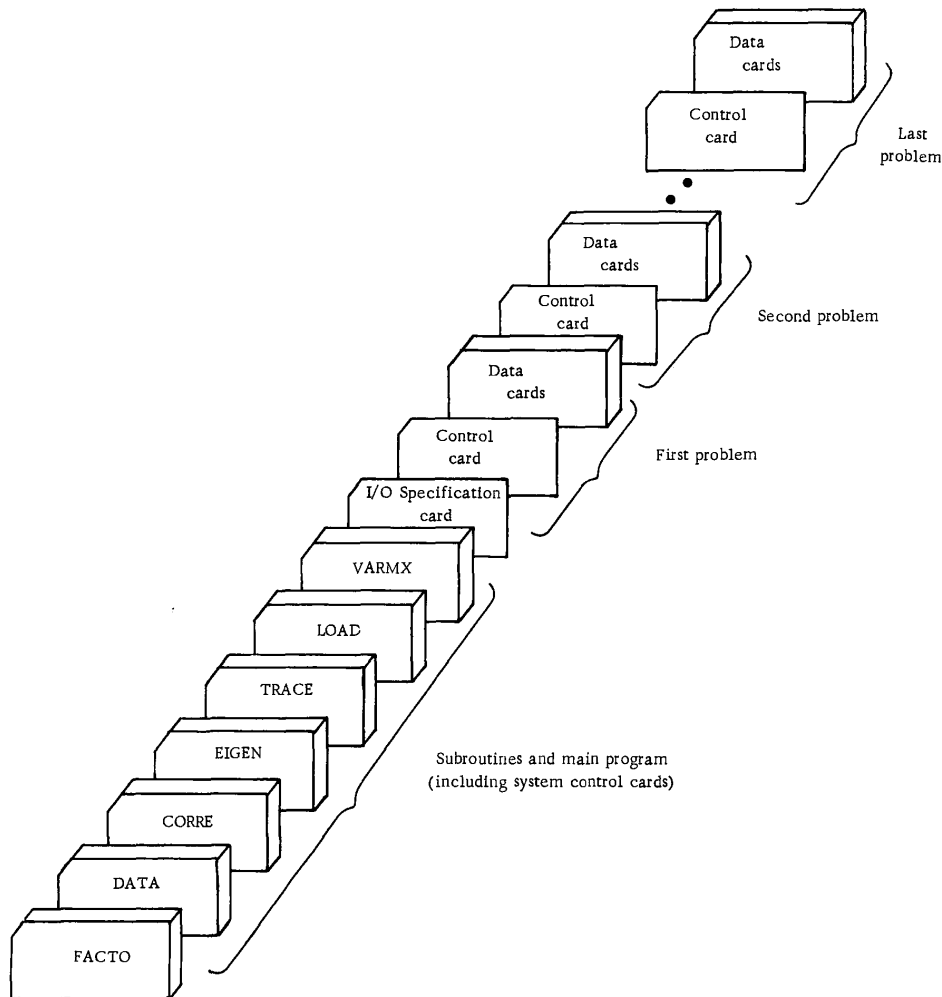


Figure 22. Deck setup (factor analysis)

## 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  $81 = 9 \times 9$ .
  - c. The dimension of array R must be greater than  $\frac{(m+1)m}{2}$

```
FACTO ANALYSIS.....SAMPLE
NO. OF CASES      23
NO. OF VARIABLES   9

MEANS
  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000
  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000

STANDARD DEVIATIONS
  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000

CORRELATION COEFFICIENTS
  1.00000  0.34866  0.11874  0.12101  0.21917  0.03954  0.22001  0.11268  0.00017
  0.34866  1.00000  0.41511  0.15572  0.08244  0.07622  0.03204  0.03397  0.00000
  0.11874  0.41511  1.00000  0.41511  0.03176  0.03176  0.03176  0.03176  0.03176
  0.12101  0.15572  0.41511  1.00000  0.03176  0.03176  0.03176  0.03176  0.03176
  0.21917  0.08244  0.03176  0.03176  1.00000  0.03176  0.03176  0.03176  0.03176
  0.03954  0.07622  0.03176  0.03176  0.03176  1.00000  0.03176  0.03176  0.03176
  0.22001  0.03204  0.03176  0.03176  0.03176  0.03176  1.00000  0.03176  0.03176
  0.11268  0.03397  0.03176  0.03176  0.03176  0.03176  0.03176  1.00000  0.03176
  0.00017  0.00000  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  1.00000

EIGENVALUES
  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000  1.00000

CUMULATIVE PERCENTAGE OF EIGENVALUES
  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000

EIGENVECTORS
  VECTOR 1  0.34866  0.11874  0.12101  0.21917  0.03954  0.22001  0.11268  0.00017
  VECTOR 2  0.41511  0.15572  0.08244  0.07622  0.03204  0.03397  0.00000
  VECTOR 3  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VECTOR 4  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VECTOR 5  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VECTOR 6  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VECTOR 7  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VECTOR 8  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VECTOR 9  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176

FACTOR MATRIX 1 = FACTORS1
  VARIABLE 1  0.34866  0.11874  0.12101  0.21917  0.03954  0.22001  0.11268  0.00017
  VARIABLE 2  0.41511  0.15572  0.08244  0.07622  0.03204  0.03397  0.00000
  VARIABLE 3  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VARIABLE 4  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VARIABLE 5  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VARIABLE 6  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VARIABLE 7  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VARIABLE 8  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VARIABLE 9  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176

ITERATION CYCLE
  1  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  2  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  3  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  4  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  5  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  6  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  7  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  8  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  9  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  10 0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  11 0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  12 0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000

ROTATED FACTOR MATRIX 1 = FACTORS1
  VARIABLE 1  0.34866  0.11874  0.12101  0.21917  0.03954  0.22001  0.11268  0.00017
  VARIABLE 2  0.41511  0.15572  0.08244  0.07622  0.03204  0.03397  0.00000
  VARIABLE 3  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VARIABLE 4  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VARIABLE 5  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VARIABLE 6  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VARIABLE 7  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VARIABLE 8  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176
  VARIABLE 9  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176  0.03176

CHECK ON COMMUNITIES
  VARIABLE 1  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  VARIABLE 2  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  VARIABLE 3  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  VARIABLE 4  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  VARIABLE 5  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  VARIABLE 6  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  VARIABLE 7  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  VARIABLE 8  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
  VARIABLE 9  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889  1.00000
```

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
*10LSICARD,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 NUMB OF VARIABLES M*(29)+S(29)+T(29)+XBARI(29) FACTO 3
C DIMENSION B(29)+D(29)+S(29)+T(29)+XBARI(29) FACTO 4
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE FACTO 5
C PROBLEM OF M,N,M FACTO 6
C DIMENSION V(841) FACTO 7
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO FACTO 8
C L=M+1*M/2+1 FACTO 9
C DIMENSION R(435) FACTO 10
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO 51.. FACTO 11
C DIMENSION TV(51) FACTO 12
C COMMON M,N,M FACTO 13
C 1 FORMAT(///21H FACTOR ANALYSIS.....A*+A2//3X+12HNO. OF CASES+4X FACTO 14
C 116/3X+10HNO. OF VARIABLES+16// FACTO 15
C 2 FORMAT(///6H MEANS/(8F15.5)) FACTO 16
C 3 FORMAT(///20H STANDARD DEVIATIONS/(8F15.5)) FACTO 17
C 4 FORMAT(///25H CORRELATION COEFFICIENTS) FACTO 18
C 5 FORMAT(///4H ROWS/(10F12.5)) FACTO 19
C 6 FORMAT(///12H EIGENVALUES/(10F12.5)) FACTO 20
C 7 FORMAT(///37H CUMULATIVE PERCENTAGE OF EIGENVALUES/(10F12.5)) FACTO 21
C 8 FORMAT(///13H EIGENVECTORS) FACTO 22
C 9 FORMAT(///7H VECTOR/(4/10F12.5)) FACTO 23
C 10 FORMAT(///170H FACTOR MATRIX (,13,9H FACTORS)) FACTO 24
C 11 FORMAT(///9H VARIABLE/(3/10F12.5)) FACTO 25
C 12 FORMAT(///10H ITERATION,7X, 9H VARIANCES/8H CYCLE) FACTO 26
C 13 FORMAT(///5H F20.6) FACTO 27
C 14 FORMAT(///24H ROTATED FACTOR MATRIX (,13,9H FACTORS)) FACTO 28
C 15 FORMAT(///9H VARIABLE/(3/10F12.5)) FACTO 29
C 16 FORMAT(///23H CHECK ON COMMUNALITIES//9H VARIABLE,7X,8H ORIGINAL, FACTO 30
C 112X,5H FINAL,10X,10H DIFFERENCE) FACTO 31
C 17 FORMAT(///3F10.5) FACTO 32
C 18 FORMAT(///4X+2,15,12,F6.0) FACTO 33
C 19 FORMAT(///5H ONLY,12,30H FACTOR RETAINED, NO ROTATION ) FACTO 34
C 20 FORMAT(12I2) FACTO 35
C READ(2,20)M,N,M FACTO 36
C READ(1,13)PR,PR1,N,M,CON FACTO 37
C 100 READ (MY,13)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
C WRITE (MX,13)PR,PR1,N,M FACTO 45
C 1000 FACTO 46
C 10000 FACTO 47
C CALL CORRE (N,M,10,X,XBAR,S,V,R,D,T) FACTO 48
C PRINT MEANS FACTO 49
C WRITE (MX,21)EIGEN(J),J=1,M FACTO 50
C PRINT STANDARD DEVIATIONS FACTO 51
C WRITE (MX,31)S(J),J=1,M FACTO 52
C PRINT CORRELATION COEFFICIENTS FACTO 53
C WRITE (MX,4) FACTO 54
C DO 120 I=1,M FACTO 55
C DO 110 J=1,M FACTO 56
C IF(I=J) 102, 104, 104 FACTO 57
C 102 I=I+J-J/2 FACTO 58
C GO TO 110 FACTO 59
C 104 L=J+I-1/2 FACTO 60
C 110 D(J)=R(L) FACTO 61
C 120 WRITE (MX,51),D(J),J=1,M FACTO 62
C MV=0 FACTO 63
C CALL EIGEN (R,V,M,MV) FACTO 64
C CALL TRACE (M,R,CON,K,D) FACTO 65
C PRINT EIGENVALUES FACTO 66
C DO 130 I=1,K FACTO 67
C L=I+I-1/2 FACTO 68
C 130 S(I)=R(L) FACTO 69
C WRITE (MX,61)S(I),J=1,K FACTO 70
C PRINT CUMULATIVE PERCENTAGE OF EIGENVALUES FACTO 71
C WRITE (MX,71)D(J),J=1,K FACTO 72
C PRINT EIGENVECTORS FACTO 73
C WRITE (MX,8) FACTO 74
C L=0 FACTO 75
C DO 150 J=1,K FACTO 76
C DO 140 I=1,M FACTO 77
C L=L+1 FACTO 78
C 140 D(I)=V(L) FACTO 79
C 150 WRITE (MX,91),D(I),I=1,M FACTO 80
C CALL LOAD (M,N,R,V) FACTO 81
C PRINT FACTOR MATRIX FACTO 82
C WRITE (MX,101)K FACTO 83
C DO 180 I=1,M FACTO 84
C DO 170 J=1,K FACTO 85
C L=M+J-1/2 FACTO 86
C 170 D(J)=V(L) FACTO 87
C 180 WRITE (MX,111),D(J),J=1,K FACTO 88
C IF(K-L) 185, 185, 188 FACTO 89
C 185 WRITE (MX,191)K FACTO 90
C GO TO 100 FACTO 91
C 188 CALL VARMX (M,K,V,NC,TV,B,T,U) FACTO 92
```

```

C      PRINT VARIANCES
      NV=NC+1
      WRITE (Mx,12)
      DO 190 J=1,NV
        NC=J-1
190    WRITE (Mx,13)(NC,TV(J))
C      PRINT ROTATED FACTOR MATRIX
      WRITE (Mx,14)K
      DO 220 J=1,M
        DO 210 J=1,K
          L=J*(J-1)/2+1
210    SI(J)=VEL(J)
220    WRITE (Mx,15),((S(J),J=1,K)
C      PRINT COMMUNITIES
      WRITE (Mx,16)
      DO 230 J=1,M
        WRITE (Mx,17),S(1),T(1),O(1)
230    WRITE (Mx,17),S(1),T(1),O(1)
      GO TO 100
      END
// DUP
*STORE      WS UA FACTO
// REQ FACTO 01
*LOCALFACTO,CONCRE,IGEN,TRACE,LOAD,VARMX

```

```

FACTO 93
FACTO 94
FACTO 95
FACTO 96
FACTO 97
FACTO 98
FACTO 99
FACTO100
FACTO101
FACTO102
FACTO103
FACTO104
FACTO105
FACTO106
FACTO107
FACTO108
FACTO109
FACTO110
FACTO111

```

```

1 2
SAMPLE00023090001,0
7 7 9 7 15 36 60 15 24
13 18 25 15 13 35 61 18 30
9 18 24 23 12 45 62 14 31
7 13 25 36 11 12 63 26 32
6 8 20 7 15 46 18 28 15
10 12 30 11 10 42 27 12 17
7 6 11 7 15 35 60 20 25
16 19 25 16 13 30 64 20 30
9 22 26 24 13 40 66 15 32
8 15 26 30 13 10 66 25 34
8 10 20 8 17 40 20 30 18
9 12 28 11 8 45 30 15 19
11 17 21 30 10 45 60 17 30
9 16 26 27 14 31 59 19 17
10 15 24 18 12 29 48 18 26
11 11 30 19 19 26 57 20 30
16 9 16 20 18 31 60 21 17
9 8 19 14 16 33 67 9 19
7 14 22 9 15 37 62 11 20
8 11 23 18 9 36 61 22 24
6 6 27 23 7 40 55 24 31
10 9 26 26 10 37 57 27 29
8 10 26 15 11 42 59 20 28

```

#### SAMPLE INPUT SUBROUTINE - DATA

##### PURPOSE

READ AN OBSERVATION (IN DATA VALUES) FROM INPUT DEVICE. THIS SUBROUTINE IS CALLED BY THE SUBROUTINE CONRE 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 (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)

DIMENSION D(1)

COMMON MX,MY

1 FORMAT(12F6.0)

C READ AN OBSERVATION FROM INPUT DEVICE.

READ (MY,1) D(1),I=1,M)

RETURN

END

```

DATA 1
DATA 2
DATA 3
DATA 4
DATA 5
DATA 6
DATA 7
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 sub-routine, 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).

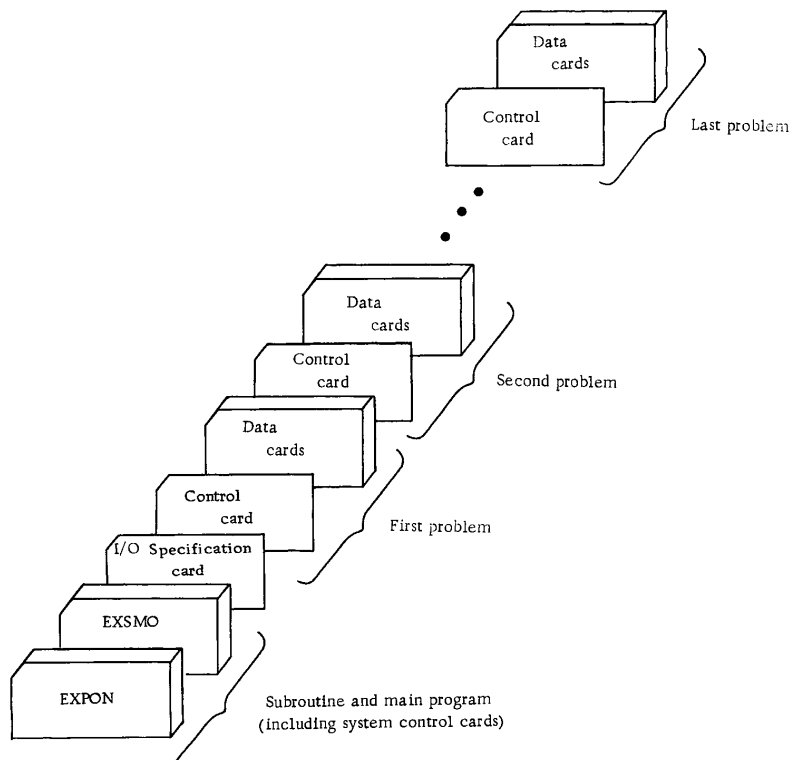


Figure 24. Deck setup (triple exponential smoothing)

# 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 |

| INPUT DATA | SMOOTHED DATA<br>(FORECAST) |
|------------|-----------------------------|
| 430.00006  | 430.00006                   |
| 426.00006  | 426.00006                   |
| 422.00006  | 422.00006                   |
| 419.00006  | 418.00006                   |
| 414.00006  | 414.29998                   |
| 413.00006  | 410.23999                   |
| 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
*IDCSICARD,TYPEWRITER,1132 PRINTER)
*NAME WORD INTEGERS
C SAMPLE MAIN PROGRAM FOR TRIPLE EXPONENTIAL SMOOTHING - EXPON
C THE FOLLOWING DIMENSION MUST BE GREATER THAN OR EQUAL TO THE
C NUMBER OF DATA POINTS IN A GIVEN TIME SERIES.
C DIMENSION X(1000),S(1000)
1 FORMAT(A4,A2,I4,F5.0,3F10.0)
2 FORMAT(12F6.0)
3 FORMAT(///34H TRIPLE EXPONENTIAL SMOOTHING.....A4,A2//22H NUMBER
1 OF DATA POINTS,I6/10H SMOOTHING CONSTANT,F9.3/)
4 FORMAT(///13H COEFFICIENTS,9A,14A,14X,14B,14X,14C)
5 FORMAT(///9H ORIGINAL,F19.5,2F15.5)
6 FORMAT(///8H UPDATED,F20.5,2F15.5/)
7 FORMAT(///27X,13HSMOOTHED DATA/7X,10HINPUT DATA,12X,10HIFORECAST))
8 FORMAT(F17.5,8X,F15.5)
9 FORMAT(2I2)
READ(2,9)MX,MY
C READ PROBLEM PARAMETER CARD
100 READ (MY,1) PR,PRI,MX,AL,A,B,C
C PR.....PROBLEM NUMBER (MAY BE ALPHAMERIC)
C PRI.....PROBLEM NUMBER (CONTINUED)
C MX.....NUMBER OF DATA POINTS IN TIME SERIES
C AL.....SMOOTHING CONSTANT
C A,B,C....COEFFICIENTS OF THE PREDICTION EQUATION
WRITE (MX,3) PR,PRI,MX,AL
C PRINT ORIGINAL COEFFICIENTS
WRITE (MX,4)
WRITE (MX,5) A,B,C
C READ TIME SERIES DATA
READ (MY,2) (X(I),I=1,MX)
CALL EXSMO (X,MX,AL,A,B,C,S)
C PRINT UPDATED COEFFICIENTS
WRITE (MX,6) A,B,C
C PRINT INPUT AND SMOOTHED DATA
WRITE (MX,7)
DO 200 I=1,MX
200 WRITE (MX,8) X(I),S(I)
GO TO 100
END
// DUP
*STORE WS UA EXPON
// REW EXPON

1 2
SAMPLE 38 0.1 0.0 0.0 0.0
430 426 422 419 414 413 412 409 411 417 422 430
438 441 447 455 461 453 448 449 454 463 470 472
476 481 483 487 491 492 485 486 482 479 479 476
472 470
```

## 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  | } | are from the Scientific Subroutine Package         |
| LOC   |   |                                                    |
| MATIN | } | are sample subroutines for matrix input and output |
| MXOUT |   |                                                    |

### 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<br>0 for general matrix<br>1 for symmetric matrix<br>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

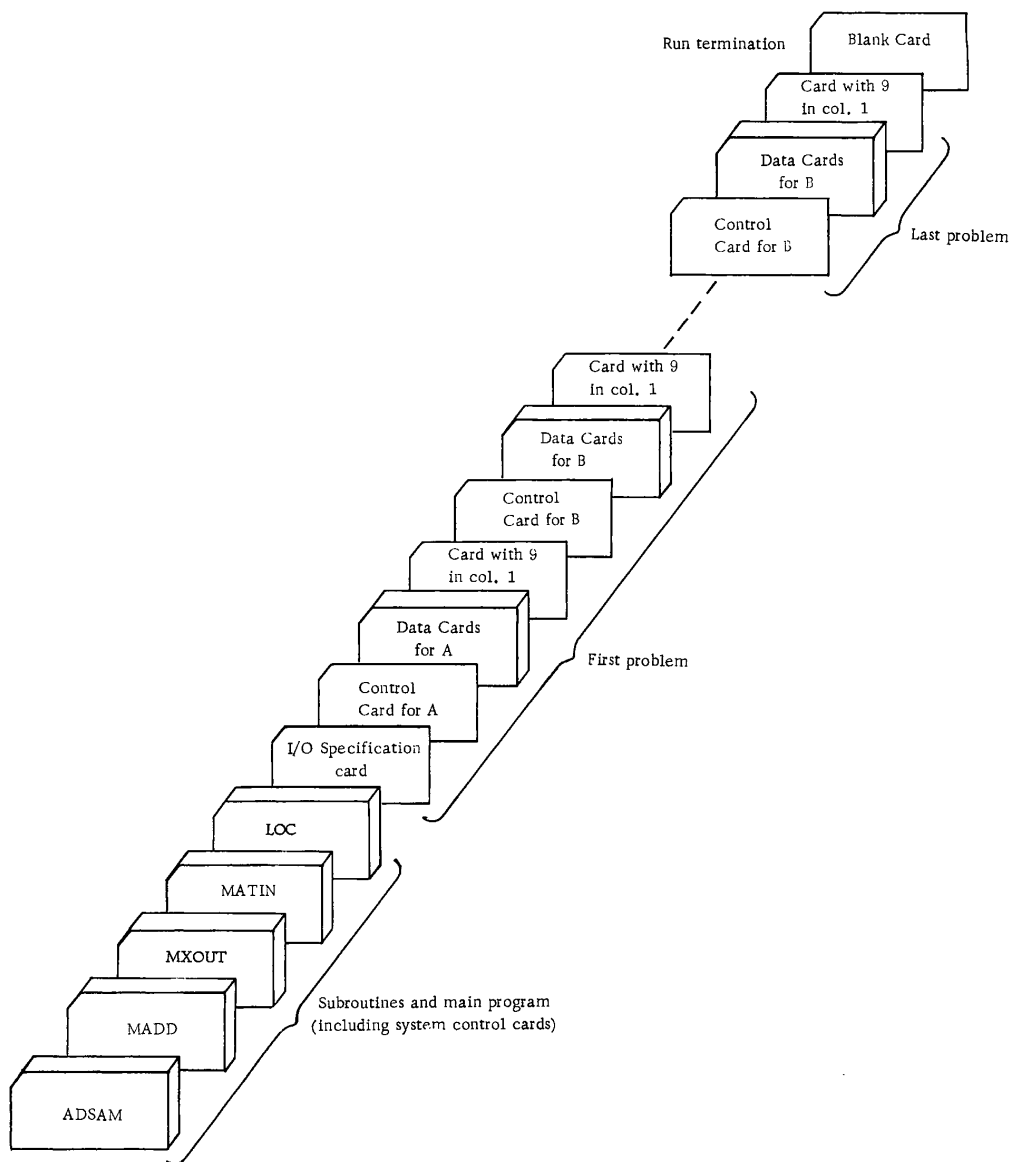


Figure 26. Deck setup (matrix addition)

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 0.708884E 00 0.409051E 00 0.314280E-02
ROW 2 0.688408E 00 0.100000E 01 0.627180E 00 0.619485E 00 0.354797E 00 0.274700E-02
ROW 3 0.100000E 01 0.688408E 00 0.760100E 00 0.790780E 00 0.438942E 00 0.332910E-02
ROW 4 0.688428E 00 0.574558E 00 0.657310E 00 0.648224E 00 0.371800E 00 0.287890E-02
ROW 5 0.744884E 00 0.614459E 00 0.702958E 00 0.686310E 00 0.397620E 00 0.297890E-02
ROW 6 0.675176E 00 0.557108E 00 0.63734E 00 0.629504E 00 0.360507E 00 0.278150E-02
ROW 7 0.892910E-02 0.274700E-02 0.314280E-02 0.314280E-02 0.314280E-02 0.177760E-02 0.140000E 01
ROW 8 0.429647E 00 0.354797E 00 0.405891E 00 0.405891E 00 0.309000E 01 0.177760E-02

MATRIX 2 8 ROWS 11 COLUMNS STORAGE MODE 0
  COLUMN 7 8 9 10 11
ROW 1 0.687640E 00 0.675176E 00 0.365591E 00 0.744884E 00 0.496326E 00
ROW 2 0.601047E 00 0.557108E 00 0.712973E 00 0.614459E 00 0.574558E 00
ROW 3 0.728478E 00 0.63734E 00 0.615202E 00 0.702958E 00 0.657310E 00
ROW 4 0.629504E 00 0.629504E 00 0.655174E 00 0.686310E 00 0.649224E 00
ROW 5 0.675176E 00 0.560507E 00 0.441109E 00 0.397620E 00 0.371800E 00
ROW 6 0.610829E 00 0.278150E-02 0.357050E-02 0.307890E-02 0.267890E-02
ROW 7 0.301100E-02 0.610829E 00 0.761287E 00 0.675176E 00 0.629504E 00
ROW 8 0.389478E 00 0.100000E 01 0.724121E 00 0.62418E 00 0.659370E 00

MATRIX 3 8 ROWS 11 COLUMNS STORAGE MODE 0
  COLUMN 1 2 3 4 5 6
ROW 1 0.750750E 00 0.619485E 00 0.708884E 00 0.100000E 01 0.408849E 00 0.318100E-02
ROW 2 0.744884E 00 0.614459E 00 0.702958E 00 0.686310E 00 0.397620E 00 0.307890E-02
ROW 3 0.686310E 00 0.712973E 00 0.615202E 00 0.695174E 00 0.648100E 00 0.357050E-02
ROW 4 0.686326E 00 0.63734E 00 0.637310E 00 0.649224E 00 0.371800E 00 0.297890E-02
ROW 5 0.675176E 00 0.557108E 00 0.63734E 00 0.629504E 00 0.360507E 00 0.278150E-02
ROW 6 0.744884E 00 0.614459E 00 0.702958E 00 0.686310E 00 0.397620E 00 0.307890E-02
ROW 7 0.686310E 00 0.712973E 00 0.615202E 00 0.695174E 00 0.648100E 00 0.357050E-02
ROW 8 0.760100E 00 0.627180E 00 0.100000E 01 0.708884E 00 0.409051E 00 0.314280E-02

MATRIX 4 8 ROWS 11 COLUMNS STORAGE MODE 0
  COLUMN 7 8 9 10 11
ROW 1 0.679201E 00 0.724121E 00 0.100000E 01 0.798884E 00 0.744884E 00
ROW 2 0.675176E 00 0.62418E 00 0.798884E 00 0.100000E 01 0.639776E 00
ROW 3 0.701287E 00 0.658270E 00 0.744884E 00 0.649797E 00 0.100000E 01
ROW 4 0.629504E 00 0.100000E 01 0.688408E 00 0.760100E 00 0.750750E 00
ROW 5 0.610829E 00 0.728478E 00 0.601047E 00 0.687640E 00 0.679201E 00
ROW 6 0.675176E 00 0.629504E 00 0.354797E 00 0.405891E 00 0.405891E 00
ROW 7 0.761287E 00 0.760100E 00 0.627180E 00 0.100000E 01 0.708884E 00
ROW 8 0.687640E 00 0.686326E 00 0.674559E 00 0.657310E 00 0.649224E 00

MATRIX 5 8 ROWS 11 COLUMNS STORAGE MODE 0
  COLUMN 1 2 3 4 5 6
ROW 1 0.131083E 01 0.129646E 01 0.170888E 01 0.170888E 01 0.606711E 00 0.624890E-02
ROW 2 0.140890E 01 0.161445E 01 0.139013E 01 0.131377E 01 0.752377E 00 0.582390E-02
ROW 3 0.169359E 01 0.137099E 01 0.157030E 01 0.155050E 01 0.805030E 00 0.689590E-02
ROW 4 0.139385E 01 0.114811E 01 0.131433E 01 0.129646E 01 0.743600E 00 0.575760E-02
ROW 5 0.161445E 01 0.117136E 01 0.134030E 01 0.132381E 01 0.759127E 00 0.587040E-02
ROW 6 0.141898E 01 0.117136E 01 0.134030E 01 0.132381E 01 0.759127E 00 0.587040E-02

ROW 7 0.868220E 00 0.713319E 00 0.618344E 00 0.608277E 00 0.442887E 00 0.109397E 01
ROW 8 0.119054E 01 0.693937E 00 0.140890E 01 0.110954E 01 0.140585E 01 0.492020E-02

MATRIX 6 8 ROWS 11 COLUMNS STORAGE MODE 0
  COLUMN 7 8 9 10 11
ROW 1 0.136656E 01 0.139929E 01 0.166359E 01 0.134333E 01 0.144333E 01
ROW 2 0.127480E 01 0.118152E 01 0.131124E 01 0.161445E 01 0.121936E 01
ROW 3 0.150876E 01 0.122121E 01 0.136200E 01 0.124693E 01 0.160731E 01
ROW 4 0.129929E 01 0.162950E 01 0.146958E 01 0.146441E 01 0.139977E 01
ROW 5 0.129434E 01 0.108896E 01 0.108219E 01 0.109028E 01 0.109100E 01
ROW 6 0.126434E 01 0.932734E 00 0.958277E 00 0.958277E 00 0.958277E 00
ROW 7 0.764297E 00 0.137099E 01 0.140844E 01 0.147311E 01 0.133846E 01
ROW 8 0.107642E 01 0.119632E 01 0.129888E 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
*ICDSICARD=TYPEWRITER+1132 PRINTER)
*ONE WORD INTEGERS
C SAMPLE MAIN PROGRAM FOR MATRIX ADDITION - ADSAM ADSAM 1
C MATRICES ARE DIMENSIONED FOR 1000 ELEMENTS, THEREFORE PHUDUCT ADSAM 2
C OF NUMBER OF ROWS BY NUMBER OF COLUMNS CANNOT EXCEED 1000. ADSAM 3
C DIMENSION A(1000),B(1000),R(1000) 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(2,1)MX,MY ADSAM 14
WRITE(MX,10) ADSAM 15
20 CALL MATIN(ICODA,A, 100,NA,MA,MSA,IER) ADSAM 16
IF(NA.NE.25)GO TO 25 ADSAM 17
25 IF(IER.NE.1)GO TO 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(ICODR,B, 100,NB,MB,MSB,IER) ADSAM 25
IF(IER.NE.1)GO TO 55 ADSAM 26
50 WRITE(MX,11)ICODR ADSAM 27
WRITE(MX,15) ADSAM 28
GO TO 20 ADSAM 29
55 WRITE(MX,14) ICODR 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(ICODR,B,NB,MB,MSB,60,120,2) ADSAM 37
ICODR=ICODA+ICODR ADSAM 38
CALL MADD(A,B,K,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 MS UA ADSAM
// REQ ADSAM
1 2 00010008001100 1
0.7601008 0.6271802 1.0000000 0.7086843 0.4058519 0.0031426 0.6876602 2
0.6751766 0.6635910 0.7446845 0.6963269 3
0.6644085 1.0000000 0.6271802 0.6194650 0.3547574 0.0027470 0.6010878 4
0.5511068 0.7125728 0.6144597 0.5745985 5
1.0000000 0.6644085 0.7601008 0.7507505 0.4299425 0.0033291 0.7284786 6
0.6373449 0.8152021 0.7029582 0.6573101 7
0.6963269 0.5745985 0.6573101 0.6492243 0.3718001 0.0028789 0.6299642 8
0.6299425 0.8051740 0.6943108 0.6492243 0.3976204 0.0030789 0.6737132 9
0.7446845 0.6144597 0.7029582 0.6943108 0.3976204 0.0030789 0.6737132 10
0.3605070 0.4611099 0.0027915 0.6108296 11
0.6751766 0.6571068 0.6373449 0.6295047 0.3605070 0.0027915 0.6108296 12
0.6751766 0.6571068 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
00020008001100 19
0.7507505 0.6194650 0.7086843 1.0000000 0.4008593 0.0031039 0.6792011 20
0.7241215 1.0000000 0.7986682 0.7468050 21
0.7446845 0.6144597 0.7029582 0.6943108 0.3976204 0.0030789 0.6737132 22
0.6244183 0.7986682 1.0000000 0.6439786 23
0.8635910 0.7125728 0.8152021 0.8051740 0.4611099 0.0035705 0.7812874 24
0.5838704 0.7468050 0.6439786 1.0000000 25
0.6963269 0.5745985 0.6573101 0.6492243 0.3718001 0.0028789 0.6299642 26
1.0000000 0.6644085 0.7601008 0.7507505 0.4299425 0.0033291 0.7284786 27
0.6751766 0.5571068 0.6373449 0.6295047 0.3605070 0.0027915 0.6108296 28
0.7284786 0.6010878 0.6876602 0.6792011 0.3889673 0.0030119 1.0000000 29
0.7446845 0.6144597 0.7029582 0.6943108 0.3976204 0.0030789 0.6737132 30
0.4299425 0.3547574 0.4058519 0.4008593 1.0000000 0.0017776 0.3889673 31
0.6635910 0.7125728 0.6144597 0.5745985 0.4058519 0.0031426 0.6876602 32
0.7601008 0.6271802 1.0000000 0.7086843 0.4058519 0.0031426 0.6876602 33
0.7601008 0.6271802 1.0000000 0.7086843 0.4058519 0.0031426 0.6876602 34
0.6963269 0.5745985 0.6573101 0.6492243 0.3718001 0.0028789 0.6299642 35
0.0027915 0.0035705 0.0030789 0.0028789 36
0.0027915 0.0035705 0.0030789 0.0028789 37
0.0027915 0.0035705 0.0030789 0.0028789 38
```

### SUBROUTINE MATIN

#### PURPOSE

READS CONTROL CARD AND MATRIX DATA ELEMENTS FROM LOGICAL UNIT 5

#### USAGE

CALL MATIN(ICODE,A,ISIZE,IR0W,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  
IR0W-UPON RETURN, IR0W 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, IR0W, 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(16,2I4,12) MATIN 6
3 FORMAT(11) MATIN 7
10C=7 MATIN 8
IER=0 MATIN 9
READ(5,2)ICODE,IR0W,ICOL,IS MATIN 10
CALL LOC(IR0W,ICOL,ICNT,IR0W,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 ICOL=ICOL MATIN 15
10C=ICOL MATIN 16
C COMPUTE NUMBER OF CARDS FOR THIS ROW MATIN 17
11 IRCDS=(ICOL-1)/10C+1 MATIN 18
IF(15-11)5,15,12 MATIN 19
12 IRCDS=1 MATIN 20
C SET UP LOOP FOR NUMBER OF CARDS IN ROW MATIN 21
15 DO 11 K=1,IRCDS MATIN 22
READ(5,1)(CARD(I),I=1,10C) MATIN 23
SKIP THROUGH DATA CARDS IF INPUT AREA TOO SMALL MATIN 24
IF(1)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)*10C+ICOL-10C+1 MATIN 28
```

### SUBROUTINE MXOUT

#### PURPOSE

PRODUCES AN OUTPUT LISTING OF ANY SIZED ARRAY ON LOGICAL UNIT 1

```

USAGE
CALL MXOUT(ICODE,A,N,M,MS,LINS,IPDS,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)
IPDS=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,IPDS,ISP)
DIMENSION A(1),B(8)
COMMON MX,MY
1 FORMAT(///5X,'MATRIX',15,6X,'13,5H ROWS,6X,'13,8H COLUMNS,
18X,'13HSTORAGE MODE',11,/)
2 FORMAT(12X,'BHCOLUMN',7(13X,'13,10X')//)
4 FORMAT(17X,'4HROW',13,7(16,6))
5 FORMAT(17X,'4HROW',13,7(16,6))
J=1
C WRITE HEADING
NEND=IPDS/16-1
LEND = (LINS/ISP)-10
10 LSTR=1
20 WRITE(MX,1)(ICODE,N,M,MS
JNT=J+NEND-1
IF(JNT-M)33,33,32
32 JNT=M
33 CONTINUE
WRITE(MX,2)(JCUR,JCUR=J,JNT)
LEND = LSTR+LEND-1
DO 30 L=LSTR,LEND
C FORM OUTPUT ROW LINE
DO 55 K=1,MFND
KK=K
JT = J+K-1
CALL LOC(L,JT,IJY,N,M,MS)
B(K)=0,0
IF(IJNT)50,50,45
45 B(K)=A(IJNT)
50 CONTINUE
C CHECK IF LAST COLUMN. IF YES GO TO 60
IF(JT-M) 55,60,60
55 CONTINUE
C END OF LINE, NOW WRITE
60 IF(ISP-1)65,65,70
65 WRITE(MX,4)(L,B(JM),JM=1,KK)
GO TO 75
70 WRITE(MX,5)(L,B(JM),JM=1,KK)
IF END OF ROWS,GO CHECK COLUMNS
75 IF(N-L)85,85,80
80 CONTINUE
C WRITE NEW HEADING
LSTR=LSTR+LEND
GO TO 20
C END OF COLUMNS, THEN RETURN
85 IF(JT-M)90,95,95
90 J=JT+1
GO TO 10
95 RETURN
END

```

## 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.

## 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.

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 0.19999998E 01 0.39999995E 01 0.59999991E 01 0.79999990E 01 0.99999981E 01
0.11999999E 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 0.14999995E 01 0.39999995E 01 0.74999995E 01 0.11999998E 02 0.17499996E 02
0.23999996E 02 0.31499992E 02 0.39999992E 02 0.49499984E 02
```

Figure 29. Output listing

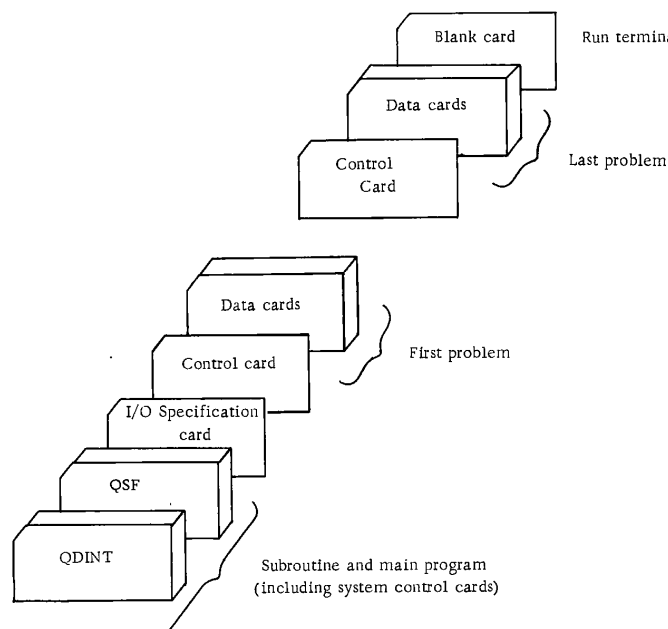


Figure 28. Deck setup (numerical quadrature integration)

## 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
*QDSC(CARD,TYPEWRITER,1132,PRINT)
*ONE WORD INTEGERS
C SAMPLE PROGRAM FOR INTEGRATION OF A TABULATED FUNCTION BY QDINT 1
C NUMERICAL QUADRATURE - 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
C DIMENSION Z(500) QDINT 5
10 FORMAT (2I5,F10.0) QDINT 6
20 FORMAT (//////) INTEGRATION OF TABULATED VALUES FOR UYUX USING SUBQDINTM01
ROUTINE QSF* //11H FUNCTION .15,34,15,17H TABULATED VALUES, QDINTM02
25X,10HINTERVAL =E15.8// QDINTM03
22 FORMAT(18H ILLEGAL CONDITION//) QDINT 10
23 FORMAT(16H NUMBER OF TABULATED VALUES IS LESS THAN THREE) QDINTM04
24 FORMAT(12H SPECIFIED INTERVAL IS ZERO) QDINT 12
30 FORMAT(7X, 'RESULTANT VALUE OF INTEGRAL AT EACH STEP IS',// QDINTM05
1E10, 'E15.8//) QDINTM06
31 FORMAT(2I2) QDINT 14
32 FORMAT(7F10.0) QDINT 15
READ 12,31MX,MY QDINT 16
35 REWIND,10//1CQ,NUMBR,SPACE QDINT 17
IF(1CQ+NUMBR)70,70,38 QDINT 18
```

```
70 STOP
38 WRITE(MX,20)1CQ,NUMBR,SPACE
IF(NUMBR-31CQ,50,50
50 READ(MY,32)1E11,1=1,NUMBR)
CALL QSF (SPACE,Z,NUMBR)
IF(SPACE)GO,2CQ,60
GO TO 35
100 WRITE(MX,22)
WRITE(MX,23)
READ(MY,32)1E11,1=1,NUMBR)
GO TO 35
200 WRITE(MX,22)
WRITE(MX,24)
GO TO 35
END
// DUP
*STORE *S UA QDINT
// XEQ QDINT
1 2
12345 20 1.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 1
2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2
2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 3
543 10 1.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 4
1.0 2.0 3.0 4.0 5.0 6.0 7.0 7 5
8.0 9.0 10.0 4.0 5.0 6.0 7.0 8 6
9 7
9 8
9 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                |

| <u>Columns</u> | <u>Contents</u>                                      | <u>For<br/>Sample<br/>Problem</u> |
|----------------|------------------------------------------------------|-----------------------------------|
| 21 - 30        | Step size                                            | 0.01                              |
| 31 - 35        | Number of steps required<br>between tabulated values | 10                                |
| 36 - 40        | Total number of tabulated<br>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.

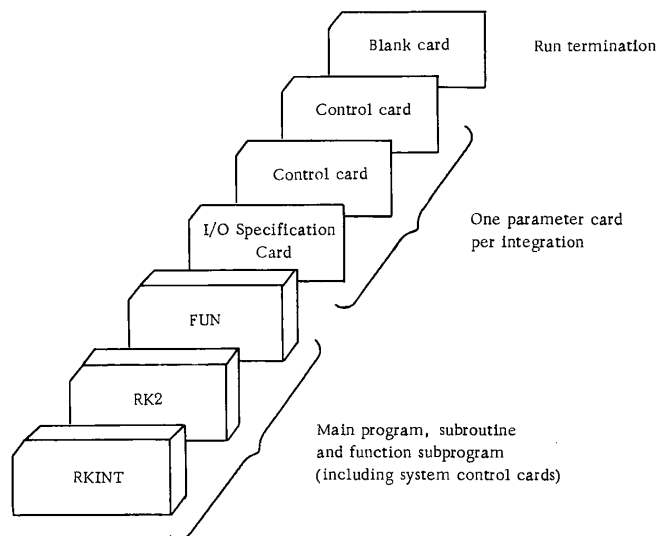


Figure 30. Deck setup (Runge-Kutta integration)

## 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

MX= 0.010 X0= 1.000 Y0= 0.000

| X    | Y(X)           |
|------|----------------|
| 1.10 | 0.95310136E+01 |
| 1.20 | 0.18232139E 00 |
| 1.30 | 0.26236997E 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

- 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
*IDC(CARD,TYPE=HITER,1132 PAINTER)
*ONE WORD INTEGERS
C SAMPLE PROGRAM FOR RUNGE-KUTTA INTEGRATION OF A GIVEN FUNCTION RKINT 1
C WITH TABULATED OUTPUT - RKINT RKINT 2
C 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
C DIMENSION A(500) RKINT 6
1 FORMAT(3F10.0,2I5) RKINT 7
2 FORMAT(////TX,***SOLUTION OF DY/DX=FUN(X,Y) BY RK2 SUBROUTINE////, RKINT 8
110X,2H//,F7.3,2X,3HNO=F7.3,2X,3HY0=F7.3////12X,14X,18X,4HY(1)/// 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)AO,YO,H,JNT,IENT RKINT 14
C CHECK IF CARD IS BLANK. IF SO, RETURN. RKINT 15
IF(IENT)30,40,20 RKINT 16
C WRITE HEADING INFORMATION. RKINT 17
20 WRITE(1,2)H,AO,YO RKINT 18
C PERFORM INTEGRATION RKINT 19
CALL RK2(FUN,H,AO,YO,JNT,IENT,A) RKINT 20
C WRITE OUTPUT RKINT 21
STEP=FLOAT(JNT)/H RKINT 22
DO 30 I=1,IENT RKINT 23
X=AO+STEP*(I-1) RKINT 24
30 WRITE(1,3)X,A(I) RKINT 25
C GO BACK AND CHECK FOR ADDITIONAL CONTROL CARD. RKINT 26
GO TO 10 RKINT 27
40 STOP RKINT 28
END RKINT 29
// DUP
*STORE HS UA RKINT
// XEQ RKINT
```

|     |     |     |     |    |    |   |
|-----|-----|-----|-----|----|----|---|
| 1 2 | 1.0 | 0.0 | .01 | 10 | 50 | 1 |
|     |     |     |     |    |    | 2 |
|     |     |     |     |    |    | 3 |

|                   |     |   |
|-------------------|-----|---|
| 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:

| <u>Columns</u> | <u>Contents</u>                      | <u>For Sample Problem</u> |
|----------------|--------------------------------------|---------------------------|
| 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.

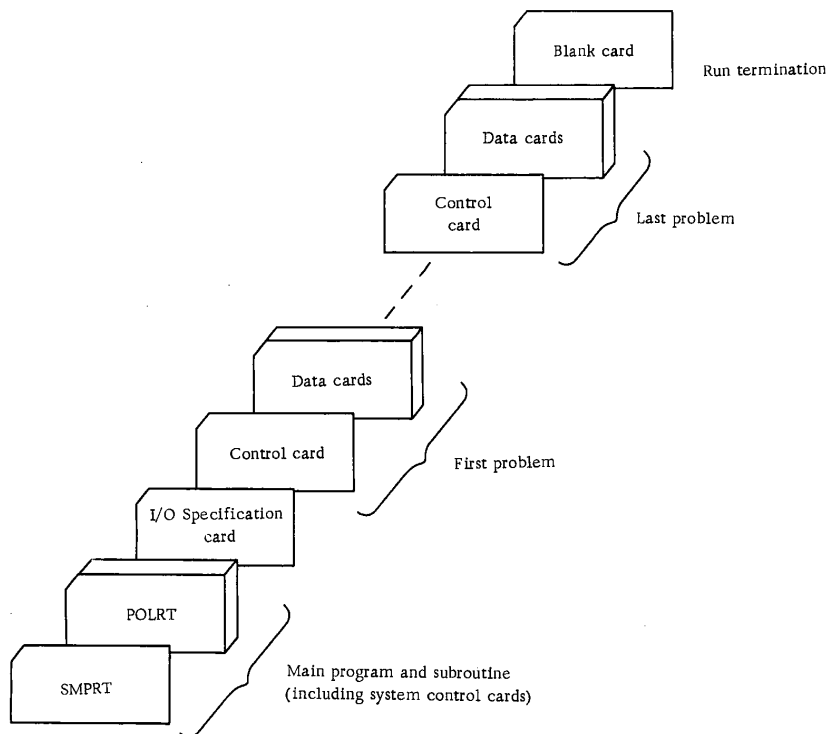


Figure 32. Deck setup (polynomial roots)

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
-0.1000000E 01  0.0000000E 00  0.0000000E 00  0.0000000E 00  0.0000000E 00  0.0000000E 00  0.0000000E 00
 0.1000000E 01  0.0000000E 00  0.0000000E 00  0.0000000E 00  0.0000000E 00  0.0000000E 00  0.0000000E 00

REAL ROOT      COMPLEX ROOT
0.2988480E 00  0.1004528E 01
0.2988480E 00  -0.1004528E 01
-0.1019270E 01  0.2436272E 00
-0.1019270E 01  -0.2436272E 00
0.9105258E 00  0.0000000E 00
0.7206227E 00  -0.7609007E 00
0.7206227E 00  0.7609007E 00
-0.4552629E 00  -0.7855384E 00
-0.4552629E 00  0.7855384E 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

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.

|       |                                                         |
|-------|---------------------------------------------------------|
| MATIN | } are sample subroutines for matrix<br>input and output |
| MXOUT |                                                         |

## 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:

```

00 // FOR
01 //IOCS(CARD,TYPE=INTER,113Z PRINTER)
02 //ONE WORD INTEGERS
03 C      SAMPLE PARAGRAM FOR KEAL AND COMPLEX ROOTS OF A KEAL POLY-
04      NAME= SMPRT
05      DIMENSION A(37),W(37),ROOT(137),RQUT(137)
06      FORMAT(1X,14,3X,12)
07      DO 90 I=1,220 KEAL AND COMPLEX ROOTS OF A POLYNOMIAL USING SUBROUTINE
08      ITINE POLY(1,1)/7TH FOR POLYNOMIAL 1,4+2X,10HOF ORDER 1,12/27TH THE SMPRT
09      INPUT COEFFICIENTS ARE//
10      40 FORMAT(1X,10)
11      50 FORMAT(6E10,7)
12      65 FORMAT(1,1,34H ORDER OF POLYNOMIAL LESS THAN ONE)
13      77 FORMAT(1,1,30H ORDER OF POLYNOMIAL GREATER THAN 36)
14      78 FORMAT(1,1,33H HIGH ORDER COEFFICIENT IS ZERO)
15      85 FORMAT(1,1,35H UNABLE TO DETERMINE ROOT. THOSE ALREADY FOUND ARE)
16      95 FORMAT(1,1,5X,9KEAL ROOT,6X,12HCOMPLEX ROOT//)
17      97 FORMAT(2E10,7)
18      98 FORMAT(2I2)
19      READ(2,3X)M,N,Y
20      READING=1,010,1,0
21      IF(10)100,100,100,20
22      WRITE(M,N,30I0,1,0)
23      J=1000+1
24      READING=N,011(1),1,1,J
25      WRITE(M,N,50I(1),1,1,J)
26      CALL POLYRT(M,Y,100,RQUT,RQUT,1,IER)
27      IF(1ER=190,60,7C
28      60 WRITE(M,N,4X,65)
29      GO TO 3
30      70 IF(1ER=375,80,78)
31      75 WRITE(M,N,77)
32      GO TO 5
33      76 WRITE(M,N,79)
34      GO TO 5
35      80 WRITE(M,N,85)
36      90 WRITE(M,N,95)
37      DO 96 I=1,1000
38      96 WRITE(M,N,97RQUT(1),RQUT(1))
39      GO TO 5
40      100 STOP
41      END
42
43 // DUP
44 //STORE M,N,S SMPRT

```

| <u>Columns</u> | <u>Contents</u>                                      | <u>For Sample Problem</u> |
|----------------|------------------------------------------------------|---------------------------|
| 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.

## 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 } are from the Scientific Subroutine  
LOC } Package

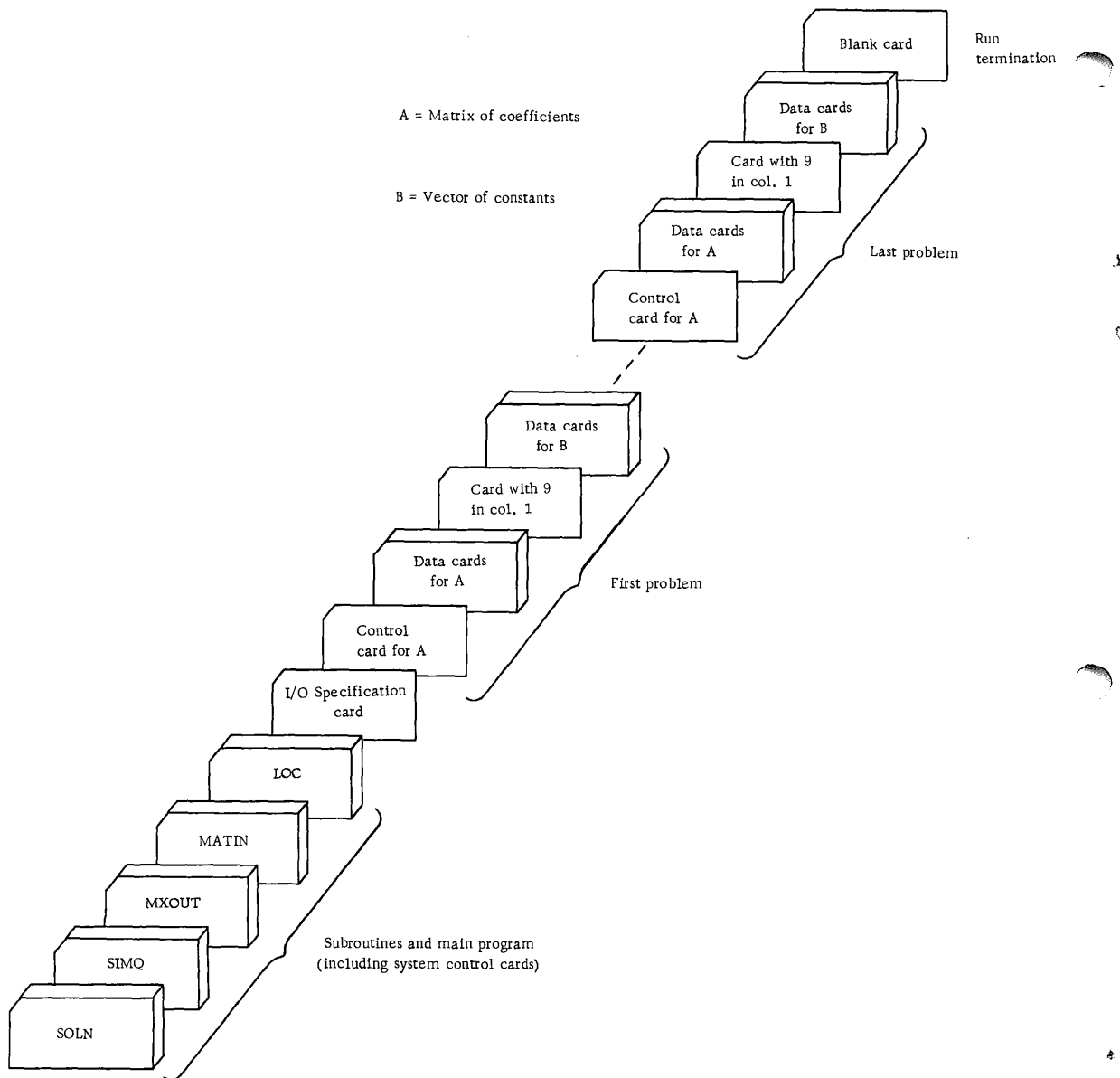


Figure 34. Deck setup (solution of simultaneous equations)

## 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  0.666666E 00  0.766100E 00  0.750750E 00  0.439942E 00  0.332910E-02
ROW 2  0.666666E 00  0.100000E 01  0.637180E 00  0.619648E 00  0.354757E 00  0.274700E-02
ROW 3  0.766100E 00  0.637180E 00  0.100000E 01  0.708684E 00  0.405951E 00  0.314260E-02
ROW 4  0.750750E 00  0.619648E 00  0.708684E 00  0.100000E 01  0.400059E 00  0.330390E-02
ROW 5  0.439942E 00  0.354757E 00  0.405951E 00  0.400059E 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  0.728678E 00  0.601087E 00  0.687660E 00  0.679201E 00  0.388967E 00  0.301190E-02
ROW 8  0.679176E 00  0.657106E 00  0.637344E 00  0.629504E 00  0.380507E 00  0.279150E-02
ROW 9  0.663991E 00  0.712972E 00  0.615202E 00  0.605174E 00  0.466109E 00  0.357050E-02
ROW 10 0.744684E 00  0.614459E 00  0.702950E 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  0.728678E 00  0.679176E 00  0.663991E 00  0.744684E 00
ROW 2  0.601087E 00  0.657106E 00  0.712972E 00  0.614459E 00
ROW 3  0.687660E 00  0.637344E 00  0.615202E 00  0.679201E 00
ROW 4  0.679176E 00  0.629504E 00  0.605174E 00  0.679201E 00
ROW 5  0.388967E 00  0.380507E 00  0.466109E 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  0.724121E 00
ROW 8  0.610829E 00  0.100000E 01  0.724121E 00  0.610829E 00
ROW 9  0.781287E 00  0.724121E 00  0.100000E 01  0.788684E 00
ROW 10 0.610829E 00  0.724121E 00  0.788684E 00  0.100000E 01

ORIGINAL B VECTOR
1      0.100000E 03
2     -0.100000E 03
3      0.100000E 02
4      0.100000E 03
5      0.100000E 02
6      0.100000E 02
7      0.100000E 02
8      0.100000E 02
9      0.100000E 02
10     0.100000E 04

SOLUTION VALUES
1      0.000000E 03
2     -0.000000E 03
3      0.000000E 02
4      0.000000E 03
5      0.000000E 02
6      0.000000E 02
7      0.000000E 02
8      0.000000E 02
9      0.000000E 02
10     0.000000E 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.

```
35 WRITE(MX,11)ICOD
GO TO 90
40 WRITE(MX,14)ICOD
GO TO 95
45 IF(N-M) 30,55,50
50 WRITE(MX,13)ICOD
GO TO 90
45 IF(M-S) 40,65,60
60 WRITE(MX,15)ICOD
GO TO 90
65 CALL MAOUT(1COD,A,N,M,S,60,120,2)
READ(MY,20)18(1),1=1,N
WRITE(MX,17)
DO 70 J=1,N
70 WRITE(MX,21)1,8(1)
CALL SIMQ(A,N,M,S)
IF(KS=1) 80,75,80
75 WRITE(MX,19)
WRITE(MX,15)
GO TO 25
80 WRITE(MX,18)
DO 85 J=1,N
85 WRITE(MX,21)1,8(1)
WRITE(MX,22)
GO TO 25
90 READ(MY,20)18(1),1=1,N
WRITE(MX,15)
GO TO 25
95 WRITE(MX,12)
STOP
END
// DUMP
// STORE WS UA SOLN
// REQ SOLN
```

```
1 2
000100100010
1.0000000 0.6644085 0.7601008 0.7507505 0.4299425 0.0033291 0.7284786
0.671766 0.8635910 0.7946845
0.6644085 1.0000000 0.6271802 0.6194650 0.3547574 0.0027470 0.6010878
0.5571068 0.7125728 0.6144597
0.7601008 0.6271802 1.0000000 0.7086843 0.4088519 0.0031426 0.6876602
0.6373449 0.6152021 0.7029582
0.7507505 0.6194650 0.7086843 1.0000000 0.4088593 0.0031039 0.6792011
0.4299425 0.8007140 0.6943108
0.4299425 0.3547574 0.4088519 0.4088593 1.0000000 0.0017776 0.3889673
0.3605070 0.4611099 0.3976204
0.0033291 0.0027470 0.0031426 0.0031039 0.0017776 1.0000000 0.0030119
0.6010878 0.0031039 0.0030119 0.0030119 0.0030119 0.0030119 1.0000000
0.7284786 0.6010878 0.6876602 0.6792011 0.3889673 0.0030119 1.0000000
0.6108296 0.7812874 0.6737132
0.671766 0.5571068 0.6373449 0.6295047 0.3605070 0.0027915 0.6108296
1.0000000 0.7241215 0.6244183
0.8635910 0.7125728 0.6152021 0.8051740 0.4611099 0.0035705 0.7812874
0.7241215 1.0000000 0.7986682
0.7446845 0.6144597 0.7029582 0.6943108 0.3976204 0.0030789 0.6737132
0.6244183 0.7986682 1.0000000
9
110.0 -120.0 10. 145. -50. 44.20 -14.
38.5 22. 1650.
26
```

##### SUBROUTINE MATIN

PURPOSE  
READS CONTROL CARD AND MATRIX DATA ELEMENTS FROM LOGICAL UNIT 5

USAGE  
CALL MATIN(ICOD,A,ISIZE,IRON,ICOL,IS,IER)

DESCRIPTION OF PARAMETERS  
ICOD-UPON RETURN, ICOD 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  
IRON-UPON RETURN, IRON 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.





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)